SYSTEM IV

User's Guide

REVISION of 8403.22

Imagination
TECHNOLOGY CORPORATION

Imagination Technology Corporation

System IV Specification and User Guide

Revised 8403.22

S4.RND/8403.22/tzm,skm

Table of Contents

TABLES & DRAWINGS

Introduction

This manual is a user guide for the Imagination Technology/Atari System IV video game hardware system. Information is provided in the following major sections:

* Overview of Features - a brief listing of capabilities
* Specifications - numbers
* System Features - an overall feature description
* Hardware/Software Interface - details necessary for programming including memory mapping and register bit assignments
* Programming - recommended practices and techniques
* Theory of Operation - grist for the hardware designer
* Hardware - component quantities and other details
* Glossary - explanation of key terms as used in this manual
* Appendices - convenient summaries

Overview of Features

This section lists the major features of System IV.

* 512 wide by 384 high pixel display area
* 1 to 8 bits per pixel in each of 2, 3, or 4 Buffers
* 16 MHz pixel display rate
* 24 KHz horizontal sweep rate
* 60 Hz vertical display rate
* Up to 256 colors using 1 Buffer plus 255 additional colors per additional Buffer
* Up to 16,777,216 selection color palette
* Independent scroll offsets for each line of each Buffer along each axis
* Physical Screen Memory surrounded by addressable Screen Space for fast clipping
* 16 MHz vector and polygon functions

    - Point plot
    - Absolute vector plot
    - Relative vector plot
    - Open polygon plot
    - Filled polygon plot

* 16 MHz character features

    - 1 megabyte stored Image Memory
    - Rectangular input windowing
    - Rectangular output clipping
    - Selectable X and/or Y direction reversal
    - Selectable X/Y transposition
    - Screen to screen transfers
    - Fractional zoom option

* Real-time Screen Memory processing during display

    - Time transparent operation
    - General erase
    - Selective erase
    - Image compression and/or restructure
    - Inter/intra Buffer merge and copy

* 8 MHz 68000 CPU
* 16MHz programmable graphic controller
* Standard speed 64K dynamic RAMs
* Multi-sourced TTL and NMOS construction
* External direct digital video input to Color Memory palette
* Copyrightable microcode and read-fuse protected PALs

Specifications

This section provides quantitative specifications about key System IV features.

CPU Address Space Allocation

Application Module Program Memory space	1 megabyte
Application Module expansion memory space	3 megabytes
Applicatin Module Scratchpad Memory Space	1 megabyte
Application Module IO space	1 megabyte
Debug Module monitor space	1 megabyte
Debug Module image emulation space	1 megabyte
Processor Module Screen Space	6 megabytes
Processor Module expansion memory space	1 megabyte
Processor Module Queue Memory and IO registers	1 megabyte

Processor Module Input-Output Registers

Queuable	
Program counter 15 bit	1
General purpose 12 bit	6
Dedicated function 8 bit	16
Control (not queuable)	3
Status (read only)	3

Speed

Horizontal segment pixel update rate	16 million per second
Pixel display erase rate	16 million per second

Asynchronous Inputs

Dedicated interrupts	4
Frame interrrupt	
Queue Processor completion	
Queue Length Register completion	
Debug Module communications	
User interrupts	3
Watchdog and power up reset	
Maximum disable event interval	8 frames (133 ms)
Power up reset latency	8 frames (133 ms)

Buffers

Buffers per system	1 to 4
Bits per pixel	1 to 8
Screen Memory horizontal offset range	0 to 511, wrapped

Screen Memory vertical offset range	0 to 511, wrapped
Visible Screen Memory size	512 x 384
Physical Screen Memory size	512 x 512
Screen Space size	4096 by 1536
Replacement Table size	256 x 8
Horizontal Offset Table size	384 x 9
Vertical Offset Table size	384 x 9

Queue System

Queue Memory size	4 to 16 K words
Next instruction fetch rate	8 MHz
Instruction deposit wait states	0
Instruction types	20+

Color Output

Color levels	256 per primary color
Color Table size	256 x 8
Color Tables	3 x 4 per primary color

CPU Memory Access Speeds

Queue Memory	0 wait states
IO Registers	
Queue Processor on	0 wait states
Queue Processor off	0+programmable
Screen Memory Write	0 wait states
Screen Memory Read	4 wait states
Program Memory	0 or programmable
Scratchpad (dynamic)	0 wait states
EEPROM	1 wait state

1 wait state = 125 ns

Graphic Controller

Cycle rate	16 MHz
Microcode width	48 bits
Microcode depth	512 instructions
Word size	12 bits
RAM size	1024 words
Branching	1, 2, 3 or 4 way
Conditionals	2 sets of 7

Display Monitor

Timing:

		unit
master clock	32.00	MHz
pixel		
frequency	16.0	MHz
period	62.5	us
horizontal		
sweep		
frequency	24.2	KHz
time	41.25	us
pixels	660	pixels
non-blanking		
time	32.0	us
pixels	512	pixels
retrace		
time	9.25	us
pixels	148	pixels
vertical		
sweep		
frequency	60.0	Hz
period	16.7	ms
pixels	266664	pixels
lines	404	lines
non-blanking		
time	15.8	ms
pixels	253440	pixels
lines	384	lines
retrace		
time	825	us
pixels	13200	pixels
lines	20	lines

input levels	
H sync	TTL
V sync	TTL
R,G,B	any linear subrange within 0 to 5 volts

System Features

Introduction

System IV is a high speed, medium resolution, bit-mapped, raster graphic, programmable arcade video game system. System IV uses up to 1 megabyte of dynamic RAM Screen Memory and up to 1 megabyte of PROM/ROM Image Memory supported by specialized hardware that simultaneously generates, displays, and erases pictures at 16 million pixels per second.

All System IV games contain the following circuit board components:

Qty	Name	Function
1	CPU	CPU, Queue Controller, system timebase
1	Update	Graphic Controller, Display system
2-4	Buffer	Screen Memory
1	Memory	Image, Program, and Scratchpad Memory
1	IO	sound effects, game controls, NVR, math processors, etc.

Only the IO board requires redesign for each new game. Since the Memory board contains all program and image data, a complete game change can be accomplished by swapping Memory and IO boards. The 6 to 8 boards listed are linked by a compact 120 conductor bus on a backplane board which also furnishes power and all other interconnect wiring. By convention, the CPU and Update boards are collectively referred to as the Processor Module (PM) and the Memory and IO boards are collectively referred to as the Application Module (AM).

The following sections describe the major System IV features in more detail.

Program Memory

System IV provides a fully decoded 1 megabyte address slot in the Memory Board for program memory. This space includes the 68000/68010 specific interrupt, trap, and reset locations. The 16 bit data path of the CPU requires a minimum of 2 byte-wide PROM/ROMs.

Scratchpad Memory

System IV provides a fully decoded 1 megabyte address slot on the Memory board for read-write memory to provide working storage for CPU stacks, queues, tables, and other purposes.

Game Input-Output

System IV provides a fully decoded 1 megabyte address slot on the IO board for game specific input-output interface with coin detection, slam switches, operator options, player controls, sound effects, and other IO hardware.

Spare Memory Space

System IV provides 3 fully decoded 1 megabyte address slots which may be utilized on either the Memory or IO boards. These may be used for additional program, scratchpad, controls, stamp and sprite generators, and any other memory mappable purpose a specific game may require. Possible future uses include non-volatile solid state memory such as EEPROM, disk drive buffer memory and interface, and three-dimensional coordinate transform hardware.

System IV Block Diagram

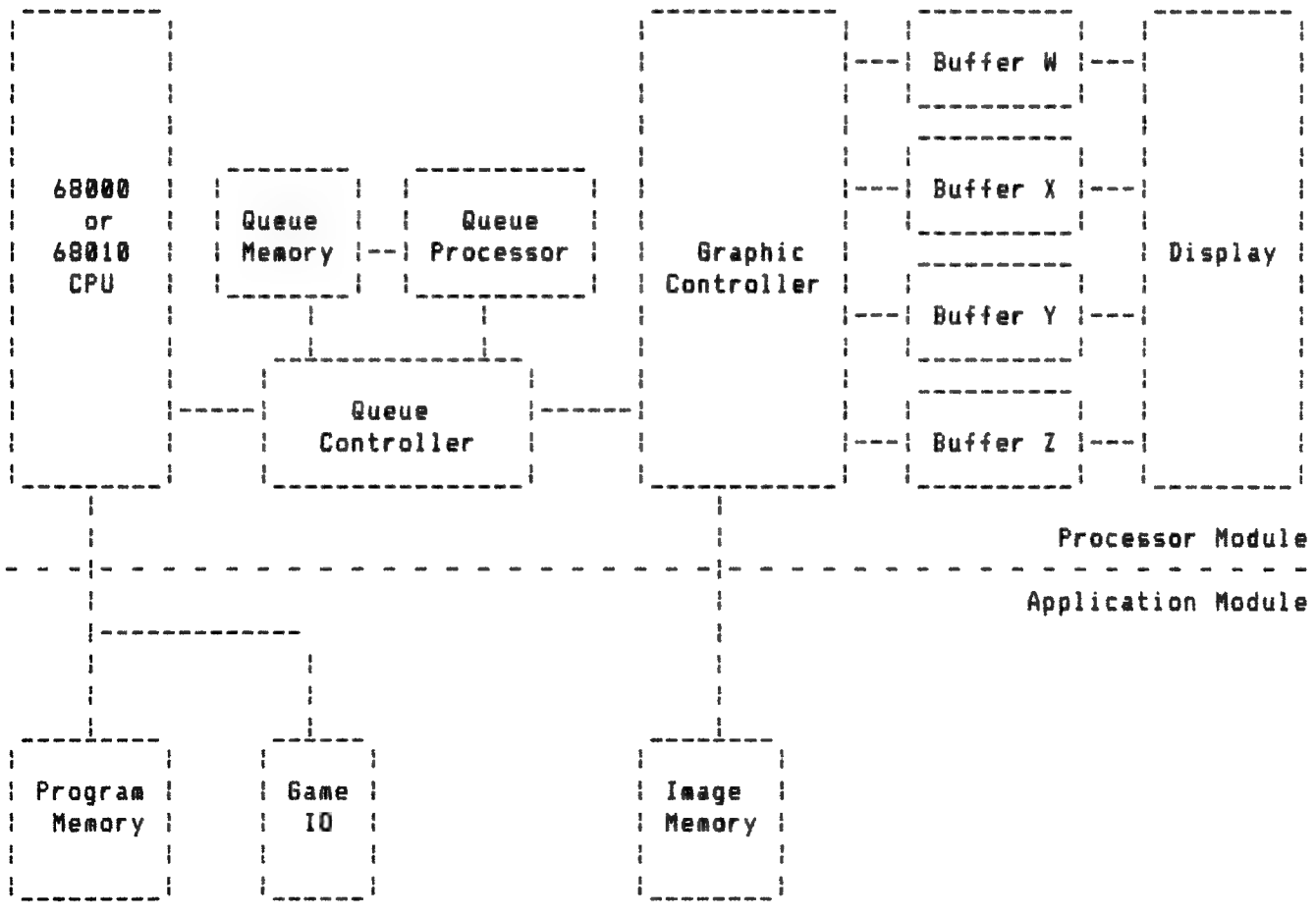


Image Memory

System IV provides up to 1 megabyte of directly addressable ROM/PROM on the Memory board for picture storage. Image Memory contents may be transferred to Screen Memory at high speed by the Graphic Controller as rectangular arrays of from 1 to 4096 pixels by from 1 to 1,048,576 pixels with up to 8 bits per pixel. Any desired rectangular subset of these picture arrays may be transferred. Although Image Memory is not located in the memory space of the CPU, the CPU may access and verify Image Memory data by first transferring the picture data to Screen Memory. Image Memory utilizes high speed sequential parallel/serial conversion and requires sub 250 ns ROM/PROM devices in multiples of 4 with a minimum complement of 4 devices.

Buffers

System IV provides up to 4 independent frame buffers that may be individually assigned to either update (generate images) or display (output images) on a frame by frame basis. Change of update/display assignment is performed during vertical retrace. Buffers are designated by the letters W, X, Y, and Z. Each Buffer contains a multipurpose 1K by 12 bit static RAM array (Translation RAM) containing lookup tables that perform Offset (unique horizontal and vertical scrolling for each line of Screen Memory), Expansion (conversion of input data for Screen Memory input), and Replacement (conversion of output data for Screen Memory input).

During update, both the CPU and the microcoded Graphic Controller create and manipulate images in Screen Memory. The 68000 CPU is linked to Screen Memory via the CPU address and data buses for direct bit-mapped read/write operation. Alternately, under immediate or deferred (i.e., via the Queue Memory) control of the CPU, the Graphic Controller operates on Screen Memory to provide high speed image transfer, vector draw, polygon fill and similar functions.

During display, any desired 384 line subset of the full 512 line Screen Memory may routed to the display monitor for viewing while additional hardware provides selective erase, modification, and Buffer prioritization.

Each Buffer has its own Buffer Control Register (BCR) which sets the Buffer's display/update assignment, enables the Buffer's Screen Memory for write, enables the Translation RAM for write, and controls the source of feedback data used by the Replacement Table. Not only do the BCRs determine which Buffers are assigned to update or display, but the BCR's make it possible to write to any desired combination of Buffers assigned to update.

Buffer Screen Memory

Each Buffer contains a single 512 wide by 512 high by from 1 to 8 bits deep memory array designated Screen Memory. To simplify programming, horizontal addressing has been extended from 9 bits (the minimum required to distinguish 512 horizontal positions) to a more leisurely 12 bits while vertical ad-

dressing has been extended to 11 bits. This extended addressing overlaps and surrounds Screen Memory with a larger area designated Screen Space. Although only the Screen Memory subset of Screen Space contains physical memory, all of the Screen Space may be written and read by both the CPU and Graphic Controller. Reading the Screen Space area outside of Screen Memory returns the value of the corresponding pixel in Screen Memory (taken modulo 512 on each axis). Writing the Screen Space area outside of Screen Memory performs no operation.

Screen Memory in each Buffer is also controlled by the 8 bit data of the Buffer's Plane Select Register (PSR). The PSR may be used to selectively enable/disable in any combination the up to 8 planes that make up a full depth Screen Memory array. This feature facilitates the division of a buffer into 2 or more overlayed groups of planes that may contain different classes of images. Use of overlays permits the Replacement Table to be used to erase all or part of the contents of designated overlay groups without affecting other overlayed groups.

Buffer Translation RAM

Each Buffer uses a 1024 by 12 bit RAM to perform address and data conversions during both Update and Display. Two 384 by 9 bit subsets are used for vertical and horizontal offset while 8 bits of each of the remaining 256 locations are used for Replacement/Expansion. These features are described in further detail below.

Buffer Offset Tables

Each Buffer contains two sets of programmable 9 bit offsets (displacements) for each of the 384 Screen Memory lines that are output during display. These sets are designated the Horizontal Offset Table and the Vertical Offset Table. The use of these tables may be enabled or disabled during update and is always enabled during display. Offset facilitates scrolling along either axis, compensates for horizontal drift during Replacement, and permits a variety of visual effects. The Vertical Offset Table makes it possible to use any 384 of the 512 available lines for display in any order desired including reuse of the same line. The Horizontal Offset Table compensates for the 4 pixel per frame rightward rotation of the Screen Memory contents caused by Replacement. The Graphic Controller provides a high speed command to fill any desired subrange of either Offset Table. The Offset Tables reside in two 384 by 9 bit subranges of the Translation RAM of each Buffer.

Buffer Expansion Table

Each Buffer contains a programmable 256 by 8 bit table to convert compressed Image Memory picture data into 8 bit wide data for Screen Memory. Compressed pictures read from Image Memory are masked, shifted, and merged by the Graphic Controller and then fed to the Expansion Table. The Expansion Table converts these 8 bit values into other 8 bit values for input to Screen Memory. This feature serves two purposes: (1) it completes the task of de-

compressing pictures in Image Memory and (2) it redefines picture elements to best suit the current data organization of the Buffer in use.

Although the Expansion Table resides in the same 256 by 8 bit subrange of the Translation RAM used by the Replacement Table, no conflict arises because Expansion is performed only during update while Replacement is performed (generally) only during display. Expansion may be enabled or disabled as required on a per Buffer basis for each update access. The Graphic Controller provides both fill and block transfer instructions for use in creating and maintaining Expansion Tables.

Buffer Replacement Table

Each Buffer contains a programmable 256 by 8 bit table that provides selective erase and/or modification of Screen Memory during display based on the contents of Screen Memory or the Screen Memory of any other Buffer assigned to display. This Replacement Table effectively doubles the time available to the CPU and Graphic Controller for image generation. In operation, Screen Memory data is routed through two 4 to 1 multiplexors into a 256 by 8 bit RAM lookup table and then back to Screen Memory input using delayed read-modify-write. Replacement may be enabled or disabled as required on a per Buffer and per frame basis using the TRW bit of the BCR of the selected Buffers. The Graphic Controller provides both fill and block transfer instructions for use in creating and maintaining the Replacement Table.

Continuous 16 MHz operation introduces a 4 pixel delay between read and write locations which causes a 4 bit wrapped shift between frames. This generally undesirable side effect is fully compensated for by appropriate programming of the Horizontal Offset Table.

The Replacement Table resides in the same 256 by 8 bit subrange of the Translation RAM used by the Expansion Table. This dual use of Translation RAM is possible because Replacement is performed during display while Expansion is performed during update.

Careful programming of the Replacement Table permits a wide variety of useful operations. For example, to erase the entire contents of a Buffer, all 256 addresses of the Replacement Table could be programmed to 0 or any other desired "erased" value. Any values to be retained are programmed by setting the data in the Replacement Table equal to the address of the desired values. Objects overlayed on background in the same Buffer may be selectively erased if the objects and background exist in separate planes. This is accomplished by programming the Replacement Table such that all addresses containing background and object are set to contain only background. Individual objects may be removed without disturbing others by programming only those objects to be erased to background (or other values) while programming the objects to be retained to their current values.

Buffer Assignment

Each Buffer is individually assignable to update or display. This permits many operating configurations to handle the requirements of varying games and play modes within games. In the simplest possible configuration, 1 Buffer may be updated during vertical retrace and displayed for the rest of the frame. Two Buffers provide a double buffered system in which one is updated while the other is displayed until the Buffers are swapped and the first is displayed while the second is updated. Four Buffers may be used in a variety of ways, including a cyclical mode in which 3 Buffers are continually displayed while a fourth is updated and rotated in a scheduled basis.

CPU

The System IV CPU may be either a 8 MHz 68000 or 68010 microprocessor as desired. The 68010 offers a superset of the 68000 with the added benefit of substantially faster multiply, divide and single instruction loops.

The CPU is used with auto-vectored interrupts, including 3 dedicated interrupts on the CPU and Update boards, 3 game specific (i.e., unreserved) interrupts on the IO board, and 1 reserved interrupt on the Debug Module. Power up and watchdog timer reset is provided after 8 ms of idle CPU activity.

All 68000 and 68010 instructions except TAS (multi-processor synchronization) are supported. The use of the STOP instruction should generally be avoided since refresh of Scratchpad dynamic memory (if installed) occurs during CPU memory cycles and because watchdog disable becomes limited to interrupt events. In place of the STOP instruction, a simple loop may be substituted.

Queue Controller

The Queue Controller interfaces the CPU, Queue Memory, the Queue Processor, the Graphic Controller, and the Buffers. At all times, the Queue Controller enables the CPU to read and write Queue Memory using either the CPU address directly or using the hardware Queue Address Pointer.

With the Queue Processor disabled, the Queue Controller provides direct CPU access to Screen Memory (bit map operation) and to the Queue Registers for manual execution of instructions by the Graphic Controller. With the Queue Processor enabled, the Queue Controller arbitrates access of Queue Memory by the CPU (high priority) and the Graphic Controller.

Queue Processor

The Queue Processor, when enabled, fetches instructions from Queue Memory for use by the Graphic Controller.

Graphic Controller

The Graphic Controller accepts intructions and register load commands from the Queue Controller either directly from the CPU or deferred via Queue Memory and the Queue Processor. Graphic Controller instructions provide the ability to:

- Set Offset Tables
- Set Expansion Tables
- Set Replacement Tables
- Set Color Tables
- Perform CPU bit-mapped operation
- Transfer Image Memory to Screen Memory
- Draw vectors
- Draw open polygons
- Draw filled polygons

The Graphic Controller is a microcoded, MSI, 16 MHz, 12 bit microcontroller with separate ALU and accumulators, 1K by 12 bit direct or indirectly accessed RAM, 32 x 12 bit immediate data, 2 independent 12 bit counters, 0, 2 or 4 way branching based on 16 conditionals, and 512 by 48 bit wide microcode.

Graphic Controller Instructions

The Graphic Controller provides many microcoded instructions that provide bit-mapped Screen Memory access as well as more sophisticated functions such as vector plot, polygon fill, and image transfer with clipping. The following listing describes a representative set of instructions that may be microcoded.

LDGSR Load Graphic Controller Status Register

[to be added]

SCLIP Set Clipping Window
RCLIP Reset Clipping Window

The SCLIP instruction sets the Graphic Controller's programmable clipping parameters to the contents of Queue Registers GR0 through GR3. The programmable clipping parameters are used by other Graphic Controller instructions to limit that active area of Screen Memory over which images produced by transfer, vector, or polygon fill can be written. The RCLIP instruction resets the programmable clipping parameters to the full 512 by 384 displayable Screen Memory area.

SOFF Set Offset
ROFF Reset Offset

The SOFF and ROFF instructions set or reset the offset enable bit of the Graphic Controller Status Register as appropriate. With offset

set, image transfer, vector, polygon fill, and bit mapped operations utilize the Offset Tables of the active update Buffers. With offset reset, the Offset Tables are bypassed. These instructions affect only update operations. Offset is always enabled during display.

SEXP Set Expansion
REXP Reset Expansion

The SEXP and REXP instructions set or reset the expansion enable bit of the Graphic Controller Status Register as appropriate. With expansion set, image transfer operations utilize the Expansion Tables of the active update Buffers. With expansion reset, the Expansion Tables are bypassed.

STPS Set Transposition
RTPS Reset Transposition

The STPS and RTPS instructions set or reset the transposition enable bit of the Graphic Controller Status Register as appropriate. With transposition set, image transfer is performed with Image Memory horizontal and vertical addresses swapped resulting in a transposed transfer to Screen Memory. With transposition reset, transfer occurs normally.

SMCM Screen Memory to Color Memory Transfer
CMSM Color Memory to Screen Memory Transfer

The SMCR and CRSM instructions copy 1 to 256 byte long blocks of data between Screen Memory of s selected Buffer and a designated Color Table of the Color Memory. The selection of Color Table is determined by the Color Table Register. The size of the block transferred and the Screen Memory X and Y coordinates are specified using 3 of the general purpose Queue Registers.

The SMCM instruction copies the contents of Screen Memory in the one Buffer designatd by the priority bits of the Display Priority Register to the one color Color RAM designated by the Color RAM write enable bits of the DPR. The starting address in the designated Color Table is specified by the data byte directly preceeding the source data in Screen Memory.

The CMSM instruction simultaneously copies the entire contents of a 256 byte block from the selected Color Table of all 3 Color RAMS to a line of Screen Memory in up to 4 Buffers in any combination. The specific action of this instruction on each Buffer depends (1) the write enable bit of the Buffer's Screen Memory (2) the setting of the Buffer's input feedback selection. The write enable determines whether any data at all will be transferred to a particular Buffer while the feedback determines which Color RAM the Buffer will received data from.

These instructions are used to initialize, maintain, and test

Color Memory and are ordinarily used only during vertical retrace. Use of these instructions outside of the vertical retrace interval will interfere with the video output to the display monitor.

FXO
FYO

[to be added]

FER

[to be added]

TRBF Translation RAM Block Fill

The TRBF instruction fills a range of Translation RAM in the designated Buffer(s) with the specified data. This command is primarily used to initialize and maintain the Offset Table of each Buffer's Translation RAM. Three general purpose Queue Registers are used to set the data value, size of the data block, and starting address in Translation RAM.

TRNL [fil expansion/replacement tables 1 to 1]

SMTR Screen Memory to Translation RAM Transfer
TRSM Translation RAM to Screen Memory Transfer

The SMTR and TRSM intructions transfer data in blocks of 1 to 1024 bytes between Screen Memory and Translation RAM. The X and Y coordinates of the start of the data block in Screen Memory, the length of the data block, and the starting address for the data block in the Expansion or Replacement Tables are stored in 4 general purpose Queue Registers. Although the Expansion/Replacement Tables are arranged as 2 non-contiguous 128 byte sets in Translation RAM, the Graphic Controller automatically adjusts as required. To accommodate this simplified operation, the Expansion/Replacement Table start address should be specified in the range of 0 to 255.

The SMTR instruction copies a block from one line of Screen Memory in the designated Buffer(s) to the Expansion/Replacement Table of the Translation RAM of the designated Buffer(s).

The TRSM instruction copies data from a specified portion of the Expansion/Replacement Table of Translation RAM of the designated Buffer(s) to a designated line in the Buffer's Screen Memory.

IMSM Image Memory to Screen Memory Transfer

The IMSM instruction copies a rectangular subset of a character image stored in Image Memory to Screen Memory. The Image Memory address high 8 bits and low 12 bits, the Screen Memory X and Y coordinates, the height and width of the source picture to be transferred, and the address offset in Image Memory between lines are stored in the 7 general purpose Queue Registers. Transposition, image shift, image mask, and full versus partial transfer are determined by the Image Format Register. Transposition, Expansion, Offset, Clipping, and Image Merge Data are determined by values previously set by other instructions to the Graphic Controller.

APP Absolute Point Plot

The APP instruction sets the Graphic Controller X and Y Point Registers to the values passed in general purpose Queue Registers GR0 and GR1 respectively and writes a pixel in the Screen Memory of the active Buffer(s) using these coordinates and the data value contained in the Graphic Controller Pixel Register.

RPP Relative Point Plot

This instruction sets the Graphic Controller X and Y Registers respectively to the sums of the respective signed offsets passed to the general purpose Queue Registers GR0 and GR1 and the respective previous contents of the Graphic Controller X and Y Registers. This instruction writes a pixel in Screen Memory of the active Buffer(s) using the new coordinates and the data value contained in the Graphic Controller Pixel Register.

AVP Absolute Vector Plot

The AVP instruction plots an absolute vector from the current Graphic Controller X and Y Register coordinates to the point specified by the general purpose registers GR2 and GR3. The value plotted is taken from the Graphic Controller Pixel Register and the Graphic Controller X and Y Registers are set to the values of the new end points.

RVP Relative Vector Plot

The RVP instruction plots an absolute vector from the current Graphic Controller X and Y Register coordinates to the point specified by the respective sums of those coordinates and the signed values specified by general purpose registers GR2 and GR3. The value plotted is taken from the Graphic Controller Pixel Register and the Graphic Controller X and Y Registers are set to the values of the new end points.

IPF Initialize Polygon Fill

The IPF instruction sets the Graphic Controller X and Y Register coordinates to the values specified by general purpose Queue Registers GR0 and GR1, plots a point in the Screen Memory of the active Buffer(s) at these coordinates using the Graphic Controller Pixel Register, and initializes the polygon fill process.

PFAV Polygon Fill Absolute Vector

The PFAV instruction adds 1 contiguous edge vector to a polygon. Parameters are similar to Absolute Vector Plot (AVP) while operation differs in that the polygon fill table is modified and no plotting actually occurs. This command should only be issued following another PFAV or IPF instruction.

CPF Polygon Fill Last Vector

The CPF instruction causes the current contents of the polygon fill table to be used to draw a filled polygon in Screen Memory of the active Buffer(s) using the data in the Graphic Controller Pixel Register.

Display Controller

[to be added]

Image Transfer Data Manipulation

System IV performs a variety of high speed modifications to image data enroute from Image Memory to Screen Memory. Data read from Image Memory is first masked using a contiguous 0 to 8 bit range and then shifted from 0 to 7 places according to the mask and shift values stored in the Image Format Register. Next, the data is merged with the 8 bit value in the Image Data Register. For each active Buffer assigned to update, the data may then pass through a 256 by 8 bit Expansion Table. Finally, the image data is placed into from 0 to 8 image planes of the Screen Memory in the designated Buffers as determined by each Buffer's 8 bit Plane Select Register.

The mask and shift feature permits pictures containing less than 8 bits per pixel to be packed together in byte-wide PROM/ROM for better storage efficiency. For example, 3 pictures stored using 4, 3, and 1 bit per pixel each may be compacted in the same byte of Image Memory. The mask and shift hardware permits only the desired picture to be extracted and ignores the other pictures during image transfer. For convenience during program development, pictures can be stored in unpacked format until either the 1 megabyte limit is reached or the product design is otherwise completed.

Hardware/Software Interface Details

Introduction

This section details the System IV software/hardware interface and provides necessary programming information. It is assumed that the reader is acquainted with microprocessor programming in general and 68000 family microprocessors in particular.

System IV utilizes the entire 16 megabytes of address space provided by its 68000 or 68010 microprocessor. The lower half of memory is used by the Application Module (AM) and Debug Module (DM) components while the upper half is used by the Processor Module (PM) components. The game-specific AM components provide six decoded 1-megabyte slots for program, scratchpad, sound effects, non-volatile RAM, IO, and any miscellaneous future requirements. Two megabytes are reserved for the DM used in game development and production testing.

The optional DM component mechanically and electrically inserts between the IO/Memory boards and the System IV backplane to provide a software/hardware debug facility that includes monitor with symbolic disassembly, serial links to a host computer, PROM programmer, and local terminal, and other useful features for creating and maintaining games.

The PM component consists of an Update board, CPU board, and 2 to 4 Buffer boards. The PM uses 6 megabytes of addressing for Screen Space, 1 megabyte of reserved memory space, and 1 megabyte for the 4K to 16K words of Queue Memory and PM Input-Output Registers. The entire memory map is graphically presented in the accompanying drawing.

System IV Game design usually need only involve the IO and Memory boards. The software task consists of creating program code and image data for the Memory board while the hardware task consists of design of the IO board containing player controls, sound effect generators, and any desired whistles and bells. The IO board interface is sufficiently flexible to permit entire subsystems capable of direct video generation to be efficiently interfaced to the CPU and color look up tables.

System IV Memory Map

Address	Module	Function
000000	AM	Program Memory including reset, trap, and interrupt vectors
....		
0FFFFF		
100000	AM	Spare
....		
1FFFFF		
200000	AM	Spare
....		
2FFFFF		
300000	AM	Spare
....		
3FFFFF		
400000	AM	Scratchpad RAM
....		
4FFFFF		
500000	AM	Game IO (coin input, player controls, operator options,
....		sound effects, non-volatile memory, etc.)
5FFFFF		
600000	DM	Reserved for development system (Monitor program, EEPROM,
....		scratchpad, UARTS, mapping RAM, miscellaneous IO)
6FFFFF		
700000	DM	Reserved for development system (Image Emulation Memory)
....		MATHBOX
7FFFFF		
800000	PM	Screen Space - read and write, byte and word
....		
DFFFFF		
E00000	PM	unused
....		DETACHED
EFFFFF		
F00000	PM	Queue Memory - 4, 8, 12, or 16 K words
....		
FF7FFF		
FF8000	PM	Queue Registers - 16 8 bit, 7 12 bit, 1 15 bit
....		synchronous load mode
FF803F		
FF8040	PM	Control Registers - 3 16 bit
....		
FF8045		
FF8400	PM	Queue Registers - 16 8 bit, 7 12 bit, 1 15 bit
....		asynchronous load mode
FF843F		
FF8800	PM	Status Registers - 2 12 bit, 1 16 bit
....		
FF8805		

CPU

The 68000 or 68010 CPU runs at 8 MHz and has access to all 16 megabytes of its address space. All CPU instructions are supported except the multiprocessor synchronization instruction TAS. Bit test, set, and clear instructions are available only on memory that is byte accessable and supports both read and write.

System IV uses the 68000/68010 interrupts in autovector mode. Levels 1, 2, and 6 are used by the PM, level 3 and 5 are available for the AM, and level 4 is reserved for the DM.

The PM interrupts provide synchronization. The level 6 interrupt reports the start of vertical retrace. This interrupt controls events that must occur during vertical retrace such as modification of the Color Memory and change of Buffer update/display assignments. This interrupt is also appropriate for advancing the software game clock. The level 6 interrupt may be hardware disabled via a bit in the Master Control Register (MCR). The level 2 interrupt reports when the Queue Length Register has reached 0 or underflowed. This interrupt simplifies checking for Queue Memory overflow when using the Queue Processor. The level 2 interrupt may be hardware disabled via a data bit in the Queue Length Register. The level 1 interrupt reports when the Queue Processor has completed operation. This is accomplished by detecting a particular address in Queue Memory. No hardware disable is provided because the level 1 interrupt may be disabled by the CPU interrupt mask without affecting other interrupts.

The DM reserved interupt is used for communication with the UARTs on the DM to perform IO to the host, terminal, and PROM programmer ports.

The AM reserved interrupts are uncommitted and may be used for any desired game-specific function.

[Discussion of wait states and hold for GC.rdy]

Interrupt Level Assignments

Level	Module	Function
1	PM	Queue Processor (FAR maximum address) completion
2	PM	Queue Length Register (QLR) completion
3	AM	Game specific
4	DM	UARTs
5	AM	Game specific
6	PM	Start of Vertical Retrace
7	PM	Unused

Bit-Mapped Operation

[give address values]

With the Queue Processor disabled (i.e., the appropriate control bit in the MCR set to 0), Screen Space (addresses 800000H through DFFFFFH containing Screen Memory) is directly accessable by the CPU for bit-mapped byte/word and read/write operations. The Screen Space access is performed by the Graphic Controller and is affected by the clipping and offset enable bits in the Graphic Controller Status Register.

In all cases, CPU address bits A00 through A11 (A00 is externally extracted from UDS) are used as the horizontal coordinate while address bits A12 through A23 are used as the vertical coordinate. All Buffers that are write-enabled and set to update are written in parallel but only one Buffer (determined by 2 select bits in the MCR) is read. The exact effects of a read or write on each Buffer are determined by the contents of the Buffer's BCR, PSR, and Translation RAM as well as the clipping and offset bits of the Graphic Controller Status Register.

Screen Space

Screen Memory is surrounded by addressable but unoccupied memory space to simplify clipping which behaves like "write only memory" when written and multiply mapped Screen Memory when read. The address space allocated for this write-only memory together with the read/write Screen Memory is designated the Screen Space.

Screen Space forms a rectangular array 1000H (4096) wide by 600H (1536) high that occupies 6 megabytes starting at CPU address 800000H. During bit mapped operation the low 12 bits of the CPU address range from 000H through FFFH (4095) representing X coordinates 000H through FFFH (4095). The next 12 CPU address bits range from from 800H (2048) through DFFH (3583) representing Y coordinates 000H through 5FFH (1536). Screen Memory occupies a 200H (512) by 200H (512) subset of Screen Space starting at A00800H.

Most System IV operations are performed using either a Screen Space or Screen Memory coordinate system. In a Screen Space system, (0,0) is the upper left corner of the 1000H (4096) by 600H (1536) Screen Space and (200H,800H) is the upper left corner of Screen Memory. In a Screen Memory system, (0,0) is the upper left corner of Screen Memory and additional Screen Space is unavailable. The following examples demonstrate the conversion of coordinates (X,Y) to absolute CPU address A for each coordinate system. Both examples assume ascending X values run left to right and ascending Y values run top to bottom.

Relative Screen Space Coordinate to Absolute Screen Space Address Conversion

```
A = Y * 1000H           ; shift Y left 12 places
A = A + 800000H         ; add upper left Screen Space base address
A = A + X               ; add X
```

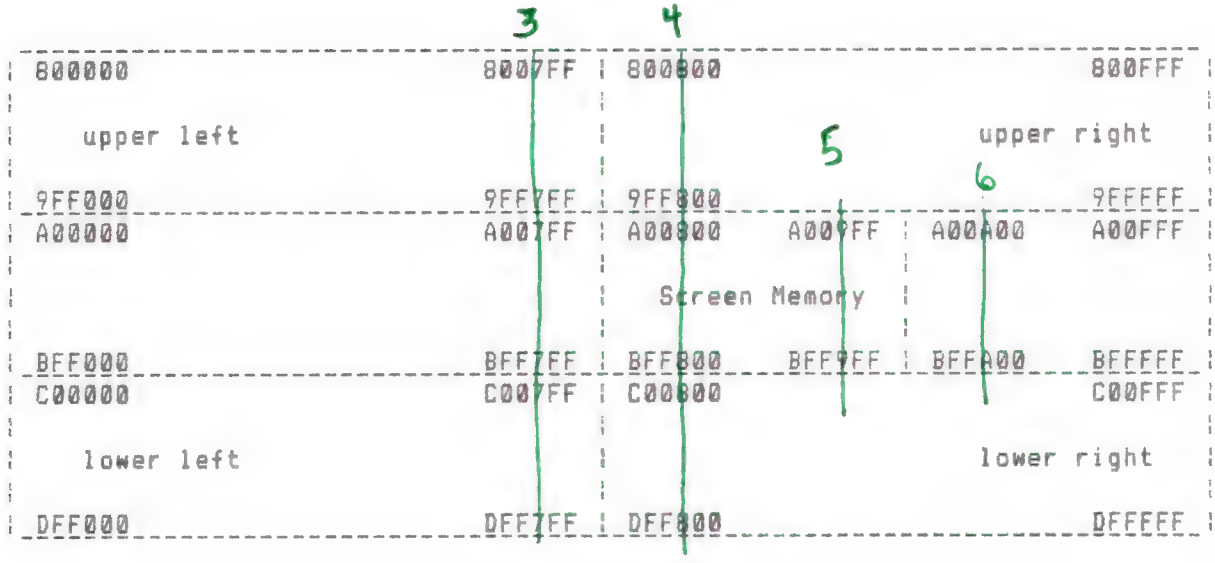
Valid for 000H <= X <= FFFH (4095) and 000H <= Y <= 5FFH (1536)

Relative Screen Memory Coordinate to Absolute Screen Space Address Conversion

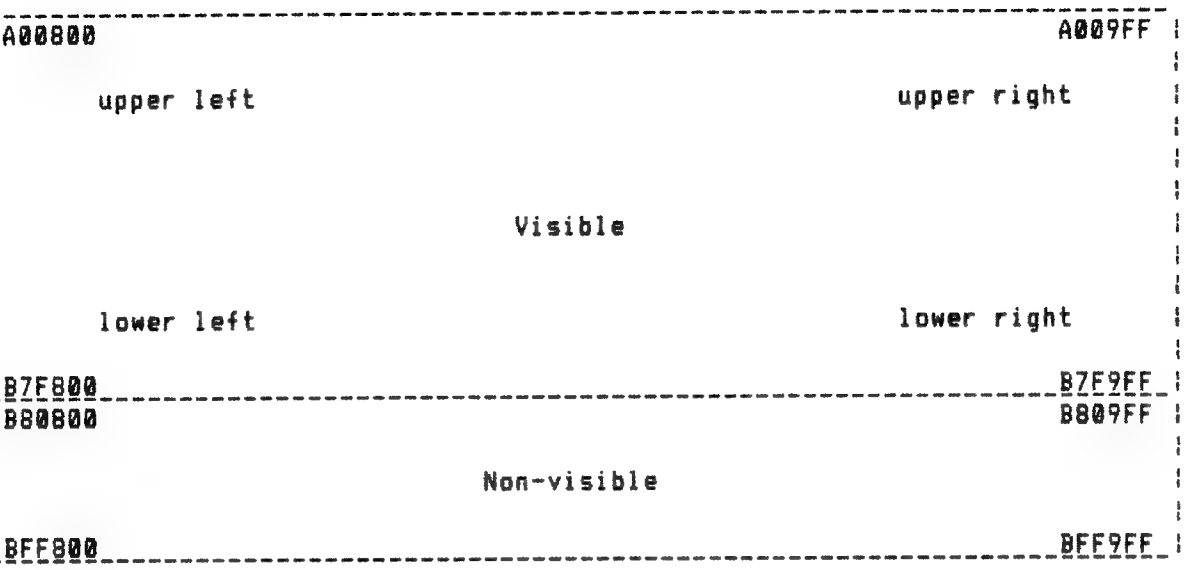
```
A = Y * 1000H           ; shift Y left 12 places
A = A + A00800H         ; add upper left Screen Memory base address
A = A + X               ; add X
```

Valid for 000H <= X <= 1FFH (511) and 000 <= Y <= 1FFH

Layout of Screen Space



Layout of Screen Memory



Translation RAM

Each Buffer has its own Translation RAM containing unique Offset and Expansion/Replacement Tables. The Translation RAM Address to Function Map shows how TR is internally split into 4 sections. The apparent split of the Replacement/Expansion Tables into two 128 byte sections is removed by Buffer and Graphic Controller hardware providing the reverse mapping given in the following Translation RAM Function to Address Map.

Translation RAM Address to Function Map

RAM Address	Contents
000H	Vertical Offset for line 000H (000)
....	
17FH	Vertical Offset for line 17FH (383)
180H	Replacement/Expansion for data 00H
....	
1FFH	Replacement/Expansion for data 7FH
200H	Horizontal offset for line 000H (000)
....	
37FH	Horizontal offset for line 17FH (383)
380H	Replacement/Expansion for data 80H
....	
3FFH	Replacement/Expansion for data FFH

Translation RAM Function to Address Map

	Screen Line	Storage Address
Vertical Offset	000H (000)	000H
	.....	....
	17FH (383)	17FH
Horizontal Offset	000H (000)	200H
	.....	....
	17FH (383)	37FH
	Input Data	Storage Address
Expansion/Replacement	00H	180H
	...	....
	7FH	1FFH
	80H	380H
	...	....
	FFH	3FFH

Offset Tables

During both update and display, the Vertical Offset Table and Horizontal Offset Table provide individual offsets for each line of Screen Memory. These offsets are added in modulo 512 to the horizontal and vertical coordinates provided by the update or display circuitry to produce the actual address supplied to Screen RAM. Certain Graphic Controller instructions bypass the Offset Tables. Setting all values in both Offset Tables to zero neutralizes the offset feature. The Offset Tables make it possible to scroll pictures along either or both axes. The Horizontal Offset Table compensates for drift induced by Replacement during display. The Vertical Offset Table can be used to invert the display for use in mirrored systems. The Offset Tables reside in a portion of a Buffer's Translation RAM.

Expansion Table

During update, this table converts the 1 to 8 bits per pixel of compressed source image data into the full available depth (up to 8 bits) of the designated Buffer for transfer to Screen Memory. The Expansion Table resides in a portion of the multipurpose Translation RAM of the corresponding Buffer. Expansion allows for efficiency in source picture storage as well as flexibility in selecting values for pixels written to Screen Memory.

Replacement Table

During display, this table converts Screen Memory output from the selected source Buffer into input data for the Screen Memory of the Buffer to which the Replacement Table belongs. This is accomplished using read-modify-write access of the Screen Memory and introduces a 4 pixel rotation to the right. The source Buffer may be chosen from among any Buffer assigned to display. The Replacement Table resides in a portion of the multipurpose Translation RAM of the corresponding Buffer.

Color Memory

During display, this 1024 by 24 bit RAM array translates the pixel from the highest priority active Buffer into any of the 16,777,216 displayable colors. Color Memory consists of three separately accessable Color RAMs for the three primary colors red, green, and blue. Each Color RAM is further divided into 4 Color Tables of 256 by 8 bits each.

Color Memory

Color Table Number	Address Range		
	Red Color RAM	Green Color RAM	Blue Color RAM
0	000H to 0FFH	000H to 0FFH	000H to 0FFH
1	100H to 1FFH	100H to 1FFH	100H to 1FFH
2	200H to 2FFH	200H to 2FFH	200H to 2FFH
3	300H to 3FFH	300H to 3FFH	300H to 3FFH

Queue Memory

Queue Memory (QM) provides 4K to 16K words of dual-ported RAM accessed by the Queue Controller for use by either the CPU or the Queue Processor. When CPU and Queue Processor requests occur simultaneously, the CPU is granted priority and always receives immediate access while the Queue Processor waits.

CPU access may be sequential or random, read or write while Queue Processor access is sequential and read only. While Queue Memory may serve as CPU scratchpad, its principal purpose is to provide a command table through which the CPU directs the Graphic Controller using the Queue Processor. This system permits both the CPU and Graphic Controller to operate independently at their maximum speeds.

The CPU port of the Queue Memory maps QM into CPU space beginning at address FF0000H. Portions of Queue Memory provide a shadow read facility for the hardware Queue Registers and Control Registers and should not be used therefore without careful consideration. The first 46H bytes of QM are used by this shadow feature.

The Queue Processor accesses Queue Memory only when the Queue Processor is enabled (via a control bit the MCR) and the Graphic Controller is ready. The Queue Processor fetches an instruction from Queue Memory and decodes it to perform one of the following operations:

* Load a Queue Register
* Execute a Graphic Controller instruction
*HPerform a Queue Processor Jump

Technically, all Queue Processor fetches load Queue Registers. An instruction is executed by loading the Execution Control Register while a jump is performed by loading the Fetch Address Register. Unlike other Queue Registers, however, loading the ECR and FAR disrupt the flow of Queue Processor fetching. The FAR may be initialized whenever the Queue Processor is off such that instruction fetches in Queue Memory will begin at the contents of the FAR when the Queue Processor is enabled.

Queue Memory Map

Address	Function
FF0000H	GR0 shadow
FF0002H	GR1 shadow
FF0004H	GR2 shadow
FF0006H	GR3 shadow
FF0008H	GR4 shadow
FF000AH	GR5 shadow
FF000CH	GR6 shadow
FF000EH	GR7 shadow
FF0010H	BCRW shadow
FF0012H	BCRX shadow
FF0014H	BCRY shadow
FF0016H	BCRZ shadow
FF0018H	PSRW shadow
FF001AH	PSRX shadow
FF001CH	PSRY shadow
FF001EH	PSRZ shadow
FF0020H	BPRW shadow
FF0022H	BPRX shadow
FF0024H	BPRY shadow
FF0026H	BPRZ shadow
FF0028H	DPR shadow
FF002AH	CTR shadow
FF002CH	IFR shadow
FF002EH	ECR shadow
FF0030H	FAR shadow
FF0032H	start of unreserved memory
....	
FF003FH	end of unreserved memory
FF0040H	MCR shadow
FF0042H	QLR shadow
FF0044H	QAR shadow
FF0046H	Start of unreserved Queue Memory
....	
FF1FFFH	end of unreserved memory with 4K word QM installed
FF2000H	
....	
FF3FFFH	end of unreserved memory with 8K word QM installed
FF4000H	
....	
FF5FFFH	end of unreserved memory with 12K word QM installed
FF6000H	
....	
FF7FFCH	end of unreserved memory with 16K word QM installed

Queue Processor

The Queue Processor reads and partially decodes instructions placed in Queue Memory by the CPU. The Queue Processor may be enabled or disabled via the <qp> bit in the MCR. During operation, the Queue Processor fetches each

instruction from Queue Memory using the current address contained in the Fetch Address Register. The instruction is decoded to load the appropriate Queue Register. A jump instruction causes a Queue Processor branch while a Graphic Controller instruction suspends further instruction fetch until it has been executed by the Graphic Controller.

When the Queue Processor is disabled (off) and the Graphic Controller is ready, all CPU write operations involving the Queue Registers are performed immediately. If the Graphic Controller is busy, the CPU access cycle is extended as necessary until the Graphic Controller becomes ready. Only then does the Queue Register operation occur and the CPU cycle terminate. The Fetch Address Register may be directly loaded while the Queue Processor is off.

When the Queue Processor is enabled (on), data written by the CPU to the Queue Register Space is combined with an operation code based on the Queue Register's address and placed in Queue Memory at the current address of the Queue Address Register (QAR). Following each instruction, the QAR is incremented and the Queue Length Register (QLR) is decremented. Meanwhile, the Queue Processor fetches previously written instructions. Most Queue Registers set data values to be used by Graphic Controller microcoded instructions. Even the execution of instructions is performed by writing to a Queue Register (the Execution Control Register (ECR)).

PM Input-Output Register Space

System IV provides 3 types of Input-Output Registers in the PM component. These are designated Queue Registers, Control Registers, and Status Registers.

The Queue Registers consist of both dedicated purpose physical hardware registers and general purpose RAM locations in the Graphic Controller and other portions of the System IV hardware. Queue Registers are unique in that they may be written directly by the CPU when the Queue Processor is off or stored in Queue Memory ("queued") when the Queue Processor is on for later transfer to hardware.

The Control Registers are 3 hardware registers that are always immediately written when accessed by the CPU.

The Status Registers are 3 input ports that provide the CPU with feedback about operation of Graphic Controller, Queue Processor, and current display address.

Queue Register Space

[give address values]

Queue Register Space is 40 bytes of CPU address space allocated to the Queue Registers (registers that may be stored in Queue Memory). Queue Register Space is used to:

* Set parameters used by the Graphic Controller
* Issue commands to the Graphic Controller
* Control the flow of instructions fetched from Queue Memory by the Queue Processor

Queue Register Space is multiply mapped in the address space of the CPU. By convention the Queue Registers are assigned to the addresses given in the Appendix. System IV provides 2 distinct methods of using the Queue Registers based on the status of the Queue Processor.

When the Queue Processor is off, data written to any Queue Register address is immediately transferred to the actual hardware device(s). Simultaneously, the appropriate operation code for the Queue Register (a function of the low 5 bits of its CPU address) is merged with the data and the result is written to a reserved portion of the Queue Memory to provide a shadow read feature. This mode is referred to as Queue Register Direct.

When the Queue Processor is on, data written to any Queue Register address is merged with the appropriate operation code for the Queue Register (the same operation code used with the Queue Processor off) and the result is written to Queue Memory at the current address of the Queue Address Register (QAR). The QAR is then incremented. The same data is also written at the same reserved address in Queue Memory used when the Queue Processor is off to maintain the shadow read feature. The Queue Register data is not transferred to hardware unless and until the Queue Processor fetches it from Queue Memory. This mode is referred to as Queue Register Deferred.

Both direct and deferred mode use the identical technique to combine CPU address with data. In each case, a 16 bit value is produced consisting of an operation code field and a operand field. The operation code is a function of CPU address bits A01 through A05 (the 5 least significant bits of the CPU word address). The operation code is a 0, 1, 4 or 8 bit value that is left justified and superimposed on the 16 data bits supplied by the CPU.

This automatic instruction "assembly" simplifies use of the Queue Processor, since the CPU need not perform any work to combine opcode with operand. Producing a command and writing it to the next sequential location in QM is reduced to simply writing the operand data to the address of the desired Queue Register.

Although the Queue Registers are write-only devices, a shadow read capability is provided. This is done by writing Queue Register data to a reserved portion of Queue Memory specified by the 5 least significant bits of the CPU word address. The contents of Queue Memory are returned to the CPU whenever a read of the associated Queue Register is requested. This shadow read feature permits each hardware register to be read, tested, and modified. This shadow read technique cannot provide reliable results under certain circumstances

that arise whenever the Queue Register and Queue Memory are independently written. For example, direct write access of the shadow space in QM will alter the shadow read data without altering the associated Queue Registers. Problems also arise following power up (QM and the Queue Registers cannot be relied upon to power up in the same state) or in the event of QM RAM device failure. Confidence in the shadow capability is easily determined by performing memory test on the QM. Note also that the shadow value returned by Queue Memory when the Queue Processor is on reflects the value last written to the Queue Register and not necessarily the current contents for the hardware Queue Register. Although this is generally the more desirable operational method, it must be realized that if the Queue Processor is disabled while unexecuted Queue Register loads remain in the Queue that the physical Queue Registers will not match their shadow values.

Control Register Space

[give address values]

Control Register Space consists of 6 bytes allocated to the 3 Control Registers. These are the Master Control Register (MCR), Queue Length Register (QLR), and Queue Address Register (QAR). Data written to a Control Register is always transferred immediately and is not affected by the status of the Queue Processor (indeed, one of the bits in the MCR is used to enable or disable the Queue Processor). A shadow read facility is provided using a reserved portion of Queue Memory to permit bit set, clear, and test.

PM Input-Output Registers

The PM Input-Output Registers are comprised of 3 Control Registers, 25 Queue Registers, and 3 Status Registers. The following sections describe each Control Register in detail. A summary is provided in the Appendix.

Graphic Registers (GR0 through GR7)

```
address FF8000 GR0
        FF8002 GR1
        FF8004 GR2
        FF8006 GR3
        FF8008 GR4
        FF800A GR5
        FF800C GR6
   type general purpose queue
 format 1<rr><d11><d10><d9><d8><d7><d6><d5><d4><d3><d2><d2><d1><d0>

        1               base opcode component for this instruction
        <rr>            register designator
                         0 = GR0
                         1 = GR1
                         2 = GR2
                         3 = GR3
                         4 = GR4
```

```
                    5 = GR5
                    6 = GR6
      <d11>..<d0>   register data
```

The 8 general purpose Graphic Registers (GR0 through GR7) place parameters directly into the Graphic Controller memory. These registers provide a variety of functions such as:

* Designate X and Y screen coordinates
* Designate Image Memory address
* Set offset between between vertically adjacent points in Image Memory
* Set height and width images for transfer
* Set clipping windows
* Set signed relative vector lengths
* Set block transfer lengths and start addresses in Translation RAM and Color Memory

Contents of the Graphic Registers are often copied by the Graphic Controller for use by Graphic Controller instructions.

Buffer Control Registers (BCRW, BCRX, BCRY, BCRZ)

```
         sync   async
address FF8010  FF8410  BCRW
        FF8012  FF8412  BCRX
        FF8014  FF8414  BCRY
        FF8016  FF8416  BCRZ

type    dedicated queue

format  111100<bb><U/D><L/R><SRW><TRW><BFH1><BFH0><BFL1><BFL0>

        111100          base opcode component for this instruction
        <bb>            Buffer designator opcode component
                          0 = Buffer W
                          1 = Buffer X
                          2 = Buffer Y
                          3 = Buffer Z
        <U/D>           update/display mode
                          0 = update mode (create images)
                          1 = display mode (show images)
        <L/R>           horizontal direction
                          0 = left to right (normal)
                          1 = right to left (reversed)
        <SRW>           Screen RAM write enable
                          0 = write disable (write protect)
                          1 = write enable
        <TRW>           Translation RAM write enable
                          0 = write disable (write protect)
                          1 = write enable
```

<BFH1><BFH0> Buffer feedback high nibble select
0 = feedback from Buffer W high nibble
1 = feedback from Buffer X high nibble
2 = feedback from Buffer Y high nibble
3 = feedback from Buffer Z high nibble

<BFL1><BFL0> Buffer feedback low nibble select
0 = feedback from Buffer W low nibble
1 = feedback from Buffer X low nibble
2 = feedback from Buffer Y low nibble
3 = feedback from Buffer Z low nibble

The Buffer Control Registers (BCRW, BCRX, BCRY, and BCRZ) are used to set modes and other operating parameters for the respective Buffers. The state of all BCR bits except bit <U/D> should only be changed while in update mode or during vertical retrace. The state of bit <U/D> should be changed only during vertical retrace. Change of state of these bits at other times may result in Screen RAM data errors or undesirable display glitches.

Bit <SRW> enables the Screen RAM of the related Buffer for writing when set (1) and disables writing when reset (0). Because both the Graphic Controller and display circuitry affect all Buffers assigned to update and display respectively, providing a separate <SRW> bit for each Buffer allows selective write control. In update mode, <SRW> determines which Buffer(s) will be written during image transfers as well as during CPU Screen Memory Direct access. In display mode, <SRW> determines which Buffer(s) will be written via Replacement.

Buffer designator bits <bb> provide the 2 lsbs of the complete 8 bit opcode for the BCR of the desired Buffer.

Bit <U/D> assigns the related Buffer to display when set (1) and to update when reset (0).

Bit <L/R> provides normal left to right horizontal pixel sequencing when reset (0) and reversed (right to left) sequencing when set. This bit is effective during both display and update but only affects operations involving consecutive, non single, pixel transfers (i.e., CPU byte writes and image transfers 1 bit wide have no left-right direction and <L/R> is irrelevent for their use).

Bit <TRW> enables the Translation RAM of the related Buffer for writing when set (1) and disables writing when reset (0). This bit should only be set to the enabled state if the related Buffer is assigned to update (i.e., when bit <U/D> is 0) and will produce undefined and probably undesirable results if set while in display mode. Because the Graphic Controller affects all Buffers assigned to update in parallel, the provision of a separate <TRW> bit in each BCR allows individual control of write operations to the Offset/Expansion/Replacement Tables of the Translation RAM of each Buffer.

Bits <BFH1> and <BFH0> select the high nibble (bits 4 through 7) from one of Buffers W through Z to be the related Buffer's feedback data source during Replacement, Buffer to Buffer transfers, and certain other operations.

Bits <BFL1> and <BFL0> select the low nibble (bits 0 through 3) from one of Buffers W through Z to be the related Buffer's feedback data source during

Replacement, Buffer to Buffer transfers, and certain other operations.

Use of a separate Buffer Control Register for each Buffer permits one or more Buffers to be used for either update or display with common addresses, data, and controls, but using individual options. Consider the following example: Buffers W, X, and Y are assigned to update while Buffer Z is assigned to display; Buffers W and X have <SRW> set; Buffers Y and Z have <SRW> reset; Buffer W has <L/R> set; and Buffers X, Y, and Z have <L/R> reset. Under these conditions, Buffer Z will display left to right without replacement while simultaneously an update image transfer would run right to left in Buffer W and left to right in Buffer X while Buffer Y would remain idle.

Plane Select Registers (PSRW, PSRX, PSRY, PSRZ)

	sync	async	
address	FF8018	FF8418	PSRW
	FF801A	FF841A	PSRX
	FF801C	FF841C	PSRY
	FF801E	FF841E	PSRZ

type dedicated queue

format 111101<bb><s7><s6><s5><s4><s3><s2><s1><s0>

111101	base opcode component for this instruction
<bb>	Buffer designator opcode component 0 = Buffer W 1 = Buffer X 2 = Buffer Y 3 = Buffer Z
<s7>..<s0>	write enables for planes 7..0 0 = enable 1 = disable

The Plane Select Registers (PSRW, PSRX, PSRY, and PSRZ) are used to enable selected image planes of the related Buffer for read/write access. Bits <s7>..<s0> provide active-low enables for image planes 7 through 0 respectively. Data written to disabled planes is disregarded while data read from disabled planes is undefined. The PSR permits partitioning of a Buffer into overlays of user definable depth.

Buffer designator bits <bb> provide the 2 lsbs of the complete 8 bit opcode for the PSR of the desired Buffer.

Buffer Priority Registers (BPRW, BPRX, BPRY, BPRZ)

```
address FF8020 BPRW
        FF8022 BPRX
        FF8024 BPRY
        FF8026 BPRZ
type    dedicated queue
format  111110<bb><d7><d6><d5><d4><d3><d2><d1><d0>

        111110          base opcode component for this instruction
        <bb>            Buffer designator opcode component
                          0 = Buffer W
                          1 = Buffer X
                          2 = Buffer Y
                          3 = Buffer Z
        <d7>..<d0>      activity threshold data
```

The Buffer Priority Registers (BPRW, BPRX, BPRY, and BPRZ) are used to set an activity threshold data value for each Buffer during display. An Buffer pixel is condidered active if its numeric value is greater than the value set in the corresponding Buffer Priority Register.

Buffer designator bits <bb> provide the 2 lsbs of the complete 8 bit opcode for the BPR of the desired Buffer.

Display Priority Register (DPR)

```
address FF8028
   type dedicated queue
 format 1111100<m1><m0><p5><p4><p3><p2><p1><p0>

        1111100         opcode for this instruction
        <m1>..<m0>      Color Table mode select code
                          0 = All Color RAMs in read mode
                          1 = Write enable Red Color RAM
                          2 = Write enable Green Color RAM
                          3 = Write enable Blue Color RAM
        <p5>..<p0>      priority select code
```

The Display Priority Register (DPR) provides bits <p5> through <p0> which select a Buffer for output to Color Memory on both a per frame and per pixel basis. The DPR also provides bits <m1> and <m0> which choose one Color RAM for write or all Color RAMs for read.

Priority bits <p5> through <p0> determine which Buffer will be used for each pixel output to Color Memory. These bits provide all 64 possible permutations of priorities for 4 Buffers. There are 4 possibilities with 1 Buffer assigned to output, 12 with 2 Buffers, and 24 each with 3 or 4 Buffers. The priority select codes are explicitly listed in Appendix A while the algorithm is described below.

Taken as a 6 bit binary number, the priority select values split into 4 sequential ranges of 16 values each. The first value in any group of 16 values is used when only 1 Buffer is assigned to display; the next 3 values are

used when 2 Buffers are assigned to display; the next 6 are used when 3 Buffers are assigned to display; and the last 6 are used when all 4 Buffers are assigned to display. The first 16 values all designate Buffer W as first priority, the next 16 set Buffer X, the third 16 set Buffer Y, and the last 16 set Buffer Z.

When only 1 Buffer is assigned to display, values will be 0 for W, 16 for X, 32 for Y and 48 for Z.

When 2 Buffers are displayable, values will be those of the first priority Buffer (0, 16, 32 or 48 for W, X, Y or Z respectively) plus an offset of 1, 2, or 3. The offset designates the position of the second priority Buffer from among the remaining 3 Buffers arranged alphabetically. For example, if X is first priority, the remaining Buffers, alphabetically ordered, are W, Y, and Z.

When 3 Buffers are displayable, values will be those of the first priority Buffer (0, 16, etc.) plus an offset of 4, 6, or 8 used to determine the second priority Buffer, plus an offset of 0 or 1 used to determine the third priority Buffer. The first offset designates the second priority Buffer from among the remaining 3 Buffers arranged alphabetically (similar to selecting 2nd priority in a 2 Buffer scheme). The second offset designates the third priority Buffer from among the remaining 2 Buffers arranged alphabetically. For example, if Y is first priority, X may be specified as second priority from among W, X and Z using a first offset of 6 and W may be specified as third priority from among W and Z using an offset of 0 for a total of 32 (for Y) plus 6 (for X) plus 0 (for W) which equals 38.

When all 4 Buffers are displayable, the algorithm is similar to the 3 Buffer with the exceptions that (1) the offsets for the second priority are 10, 12, or 14 and (2) that the fourth priority Buffer is simply the remaining Buffer.

During display, the highest priority active Buffer with non-zero data is output. When all active Buffers are zero, the highest priority Buffer is output as default.

During write operations to Color Memory, the priority bits must be set for the desired Buffer(s). This normally will be a single Buffer priority scheme (i.e., priority values of 0, 16, 32, or 48).

Mode bits <m1>..<m0> select and enable one of the three Color RAMs (red, green, or blue) of the Color Memory for writing. Writing Color Memory is normally performed only during vertical retrace to avoid interference with the video output. While enabled for writing, the Color Memory interprets any consecutive stream of Screen Memory pixels as an 8 bit address in the current Color Table followed by one or more color data values. The first color data value will be written at the 8 bit address provided and each successive color data value will be written at the next sequential address in the current Color Table. Color Table addresses are interpreted modulo 256 such that the first color data value written after address 255 will be at address 0 of the same Color Table.

A CPU word read while the Green Color RAM is write-enabled, for example, will use the high byte read from Screen Memory as the address in the Green

Color RAM and use the low byte as green color data at that address. To transfer more than 1 color data value at a time, instructions are available that provide from 1 to 256 bytes. Note that longword accesses are really two sequential but interrupted accesses of 2 consecutive pixels each and not one continuous access of 4 consecutive pixels.

Color Table Register (CTR)

```
address FF802A
   type dedicated queue
 format 1111101<z1><z0><y1><y0><x1><x0><w1><w0>
```

1111101	opcode for this instruction
<z1><z0>	Buffer Z Color Table number
<y1><y0>	Buffer Y Color Table number
<x1><x0>	Buffer X Color Table number
<w1><w0>	Buffer W Color Table number

The Color Table Register (CTR) is used to designate a Color Table for each Buffer during display or when writing to Color Memory. The highest priority active Buffer determines which 2 bit value from the CTR will be combined with the same Buffer's 8 bit pixel value to form the full 10 bit address used by Color Memory. A separate Color Table may be designated for each Buffer or the same Color Table may be used for several or all Buffers. Providing separate Color Tables allows more color choices while reusing the same Color Table for several or all Buffers retains simplicity.

Image Format Register (IFR)

```
address FF802C
   type dedicated queue
 format 1111110<p/f><s2><s1><s0><m3><m2><m1><m0>
```

1111110	opcode for this instruction
<p/f>	partial/full transfer
	0 = full transfer
	1 = partial transfer
<s2>..<s0>	image transfer shift
<m3>..<m0>	image transfer mask

The Image Format Register (IFR) sets the Image Memory Mask to the value of bits <m3> through <m0> and the Image Memory Shift to the value of bits <s2> through <s0>. The IFR sets a status bit that controls full or partial transfer during image transfer.

Mask bits <m3>..<m0> select any of the 16 possible contiguous left and right justified ranges of enabled bits from the 8 bit Image Memory. Each enabled bit position passes the Image Memory data without modification while each disabled position supplies a 0. The mask values from 0 to 8 are equivalent to the number of right justified Image Bits enabled, while mask values from 9 to 15 provide a quantity of left justified bits equal to 16 minus the mask value. Appendix D lists all mask values.

Shift bits <s2>..<s0> control the logical shift right (msb to lsb) of the Image Memory at the output of the Mask circuitry. Nonexistent image bits 8 through 14 are treated as 0's by the shift circuitry. Appendix D lists all shift values.

Full/partial bit <pf> selects full transfer when reset and partial transfer when set. Full transfer means that all non-clipped pixels will be written to Screen Memory. Partial transfer means that only Image Memory pixels of greater or equal to a specified value will be written to Screen Memory. Pixels not written are skipped over in Screen Memory.

Execution Control Register (ECR)

```
address FF802E
   type dedicated queue
 format 11111111<i7><i6><i5><i4><i3><i2><i1><i0>

        11111111        opcode for this instruction
        <i7>..<i0>      instruction microcode vector
```

The Execution Control Register (ECR) is used to load a microcode address vector into the Graphic Controller causing immediate execution of the corresponding instruction. Microcode vector is specified by bits <i7>..<i0>.

Fetch Address Register (FAR)

```
address FF8030
   type dedicated queue
 format 0<a14><a13><a12><a11><a10><a9><a8><a7><a6><a5><a4><a3><a2><a1>0

        0               opcode for this instruction
        <a14>..<a1>     Queue Memory relative word address
        0/1             reserved jump/call indicator
```

The Fetch Address Register (FAR) is the program counter of the Queue Processor. The FAR may be loaded to perform a Queue Processor jump or initialization. Bits <a14>..<a01> are loaded into the 14 least significant bits of the FAR to provide the address of the next instruction fetch.

Master Control Register (MCR)

address FF8040
type control
format <qp><wp><s1><s0><ie6>xxx

<qp>	Queue Processor enable	
	0 = disable (Queue Processor off)	
	1 = enable (Queue Processor on)	
<wp>	Wrap enable	
	0 = Screen Memory write only	
	1 = Screen Space write	
<s1><s0>	Buffer read select	
	0 = read Buffer W	
	1 = read Buffer X	
	2 = read Buffer Y	
	3 = read Buffer Z	
<ie6>	Interrupt level 6 enable	
	0 = disable/reset interrupt	
	1 = enable/set interrupt	
xxx	unimplemented bits	

The Master Control Register (MCR) is an 8 bit Control Register used to enable/disable the Queue Processor, enable/disable the level 6 interrupts, and

Bit <qp> turns the Queue Processor on and off. When set (1), Queue Register access is deferred, (i.e., converted into an instruction and placed in Queue Memory for later execution) while the Queue Processor fetches previously deferred Queue Registers. When reset (0), all Queue Register operations are direct (i.e., access of the associated hardware device occurs immediately) and the Queue Processor is disabled. select which Buffer will be read by the CPU.

[explain <wp>], [explain <s1><s0>

Bit <ie6> enables interrupt at the start of vertical retrace. The interrupt is disabled/reset on 0 and enabled/set on 1. Once the interrupt has been enabled and recognized (serviced by the CPU) it must be explicitly reset and set again in order to be reused. (The CPU external interrupt hardware is edge-triggered and is reset each time the associated interrupt enable is set to 0).

Queue Length Register (QLR)

```
address FF8042
   type control
 format <ie2><14><13><12><11><10><9><8><7><6><5><4><3><2><1><0>

        <ie2>           interrupt level 2 control
                         0 = immediate interrupt
                         1 = interrupt on count underflow
        <14>..<10>      queue length count
```

The Queue Length Register (QLR) is a presettable counter that decrements each time the QAR is used to place a Queue Register into Queue Memory. The QLR interrupts the CPU on underflow.

Queue Address Register (QAR)

```
address FF8044
   type control
 format x<a14><a13><a12><a11><a10><a9><a8><a7><a6><a5><a4><a3><a2><a1>x

        x               unimplemented bit
        <a14>..<a1>     Queue Memory auto-incrementing input pointer address
        x               unimplemented bit
```

The Queue Address Register (QAR) provides a post-auto-incrementing pointer to Queue Memory used to input Queue Register data. Bits <a14>..<a1> initialize this pointer which is used on each Queue Register write operation that occurs when the Queue Processor is on. The Queue Address Register is automatically incremented after each use.

Display Address Register (DAR)

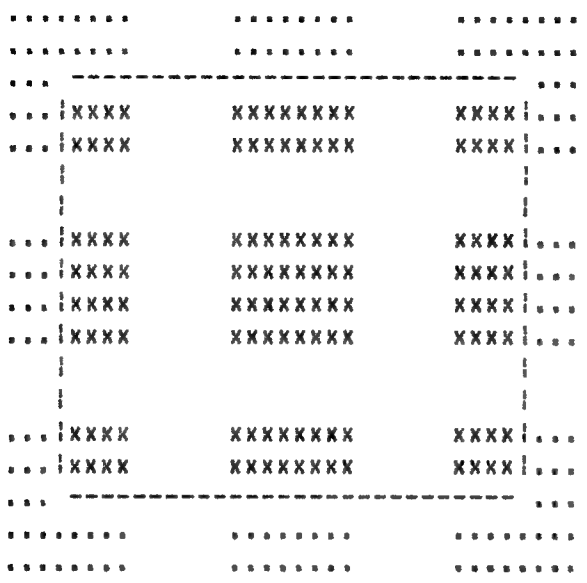
```
address FF8
   type status
 format
```

The Display Access Register (DAR) ***

Clipping

Programmable clipping limits Screen Memory operations to a specified rectangular subset. Clipping is applied relative to Screen Memory before any offset is added. The following example shows how an 8 by 5 object created by image transfer is affected by clipping to Screen Memory boundaries. The dashed lines represent the boundaries of Screen Memory. In each case, "x" represents a pixel that transferred and "." represents a pixel that was not transferred because of clipping.

Clipping During Image Transfer



Programming

Buffer Grouping

Among the primary considerations in using System IV is effective utilization of its multi-buffer architecture. The following sections describe the Buffer update/display assignment options in a fully configured 4 Buffer system.

U0/D4 - Update 0 Buffers and Display 4 Buffers

This mode achieves maximum graphic effect since it provides up to 32 bits per pixel for display. Since no Buffers are left for update, however, this mode is not used for continuous interactive update unless its update requirements are minimal enough to be completely satisfied during vertical retrace.

U1/D3 - Update 1 Buffer and Display 3 Buffers

When combined with appropriate programming of the Replacement Tables of the affected Buffers, this mode provides 3 prioritized overlays of up to 8 bits that can be collectively updated 60 times per second. Operations requiring more than 1 frame time to complete update are accommodated.

In a typical application, 3 Buffers may be individually used for foreground, midground, and background display while the 4th Buffer is used for update. Cycling the 1 update Buffer with the 3 display Buffers equally would provide a 20 Hz video rate for each display Buffer and 24 visible bits per pixel. More typically, however, the 60 frames per second could be assigned unequally to provide optimum throughput. For example, main action in midground could be updated at 40 times per second, less frequent background changes at 15 times per second, and only occasional score changes in the foreground at 5 times per second.

Sample operation of this mode showing how Buffers are cycled using Replacement is shown below.

Display Buffer Usage

Foreground	W	Z	Z	Z	Y	Y	Y	X	X	X	W	W
Midground	X	X	W	W	W	Z	Z	Z	Y	Y	Y	X
Background	Y	Y	Y	X	X	X	W	W	W	Z	Z	Z

Replacement Buffer Usage

Source	X	Y	Z	W	X	Y	Z	W	X	Y	Z	W
Target	W	X	Y	Z	W	X	Y	Z	W	X	Y	Z

Update Buffer Usage

Foreground	Z			Y			X			W		
Midground		W			Z			Y			X	
Background			X			W			Z			Y

Frame number	0	1	2	3	4	5	6	7	8	9	10	11

¦---------- time ---------->

U2/D2 - Update 2 Buffers and Display 2 Buffers

This mode provides conventional double buffered operation. Two Buffers are available for both update and display at a full 60 Hz rate. Operations requiring more than 1 frame time to complete update are accommodated.

Display Buffer Usage

Foreground	W	Y	W	Y
Background	X	Z	X	Z

Replacement Buffer Usage

Source	WX	YZ	WX	YZ
Target	WX	YZ	WX	YZ

Update Buffer Usage

Foreground	Y	W	Y	W
Background	Z	X	Z	X

Frame number	0	1	2	3

¦- time ->

U3/D1 - Update 3 Buffers and Display 1 Buffer

This mode is used only in special instances such as a complete rewrite of the play field or during set-up to enter modes U1/D3 or U0/D4. Operations requiring more than 1 frame time to complete update are accommodated.

U4/D0 - Update 4 Buffers and Display 0 Buffers

This mode is used only to initialize a playfield or run diagnostics since no data is available to the display from Screen Memory. This mode can also be used during external video input.

Watchdog Timer

System IV employs a combined power-up reset and watchdog timer circuit to initialize or re-initialize system operation. The Watchdog Timer is a 4 bit counter incremented once at the end of vertical retrace during each frame. When the counter reaches 8, signals HALT- and RESET- are driven low for 8 frames (1.33ms). This resets the CPU, clears all MCR bits, and may be used to reset any game specific hardware provided on the Application Module. Once the reset operation has finished, the system may be kept functioning by clearing the Watchdog Timer at least once every 8 frames. The current disable event is the loading of an instruction into the Execution Control Register. A full powerup reset will be performed any time 8 frames elapse without a write to Queue RAM.

Power Up

Once the 68000/68010 CPU has been reset by the Watchdog Timer some important system considerations must observed before proceeding further. First, the dynamic memory must be initialized. This is a hardware requirement imposed by the IC design of the 64K RAMs which varies with each device manufacturer. For most varieties employed thus far (Intel, Fujitsu, Micron Technology, Mostek, Texas Instruments, and Toshiba), this requirement can be met by performing 8 or more accesses of each row of each device.

This requirement can also be met by either waiting for at least 1 full frame of display or update operation in all Buffers or by explicit CPU access. The former method is more automatic but might be considered time consuming in some instances while the latter is quicker but may require some additional code. Explicit wakeup can be achieved by accessing a single byte/word from any 128 consecutive Screen Memory lines 8 times. Any number of possibilities exist to make this operation more efficient, such as 4 accesses of 256 locations or 8 accesses of 512 locations or use of the image transfer circuitry on a 128 or 256 high by 1 wide rectangle.

A second consideration after power up is to avoid subsequent system resets caused by failure to reset the Watchdog Timer.

Image Memory Data Format

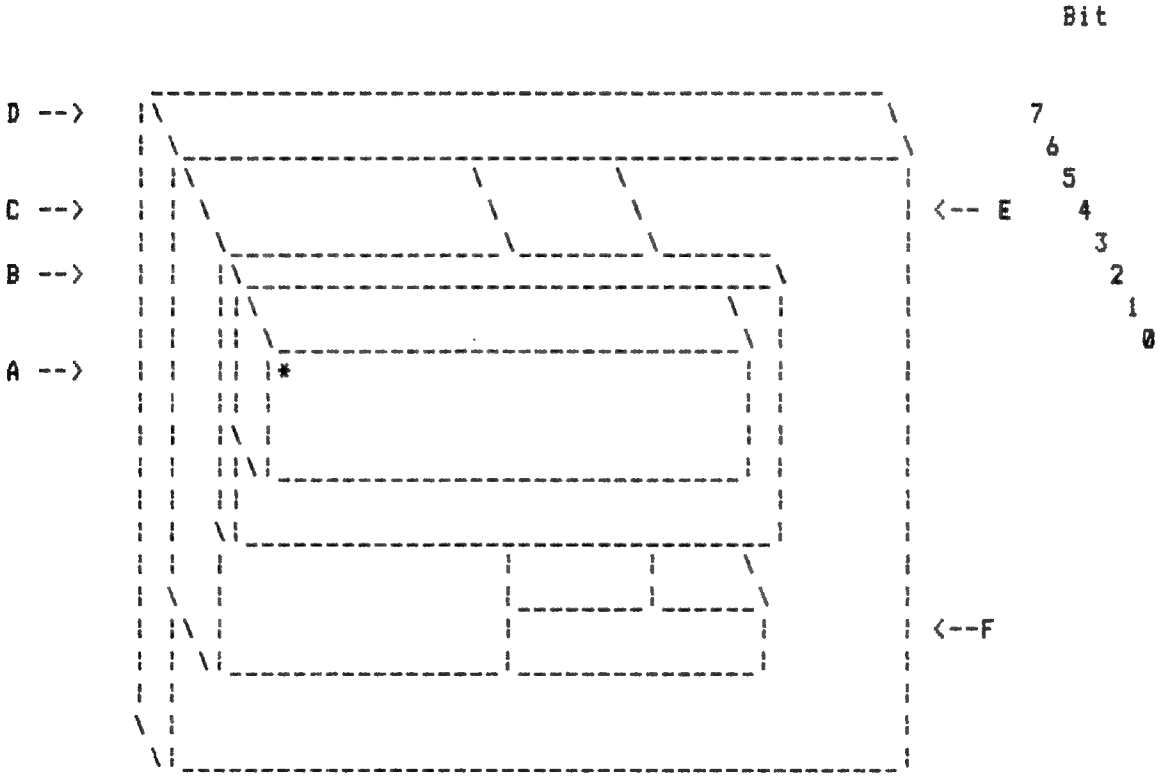
System IV uses a flexible scheme for picture storage in PROM/ROM Image Memory. All pictures are stored as rectangular arrays with from 1 to 8 bits per pixel. Each array may be from 1 to 4096 wide by 1 to 1,048,576 high. The rectangular arrays may be closely packed to make the most efficient use of

physical memory.

In the sample below, six pictures labelled A through F are shown packed together. In the figure, width and height represent picture height and width while perspective "depth" represents the 8 bits per byte organization of the Image Memory PROMs/ROMs. Note that the upper left corner of pictures A through D (under the asterisk ("*") character) coexist in the same byte of Image Memory.

During image transfer, the mask hardware is used to limit Image Memory output to only the bits in the "layers" desired. The shift hardware then moves the masked off bits into the LSB position. Using both mask and shift simplifies programming of the Expansion Table. Without the mask feature, the Expansion RAM would need to be programmed for all 256 possible 8 bit values. Without the shift feature, the same Expansion Table programming could not be used for similar pictures stored in different planes and image values would not be contiguous.

Sample Image Memory Storage



Theory of Operation

Overview

System IV simultaneously updates, displays, and erases at the rate of 16 million pixels per second using a 16 bit MOS microprocessor (CPU) and a microcoded Graphic Controller (GC). The two processors are linked via the Queue Controller (QC) either directly (for debug and slow speed operations) or via a buffered command list kept in Queue Memory (QM) and extracted by the Queue Processor (QP). One to four Buffers each provide up to 1/4 megabyte of Screen Memory (SM). A separate Image Memory (IM) provides up to 1 megabyte of PROM/ROM storage for pictures that may be transferred to SM at high speed. Each Buffer provides address offset and input data conversion tables for flexibility in addressing, erase, and other purposes.

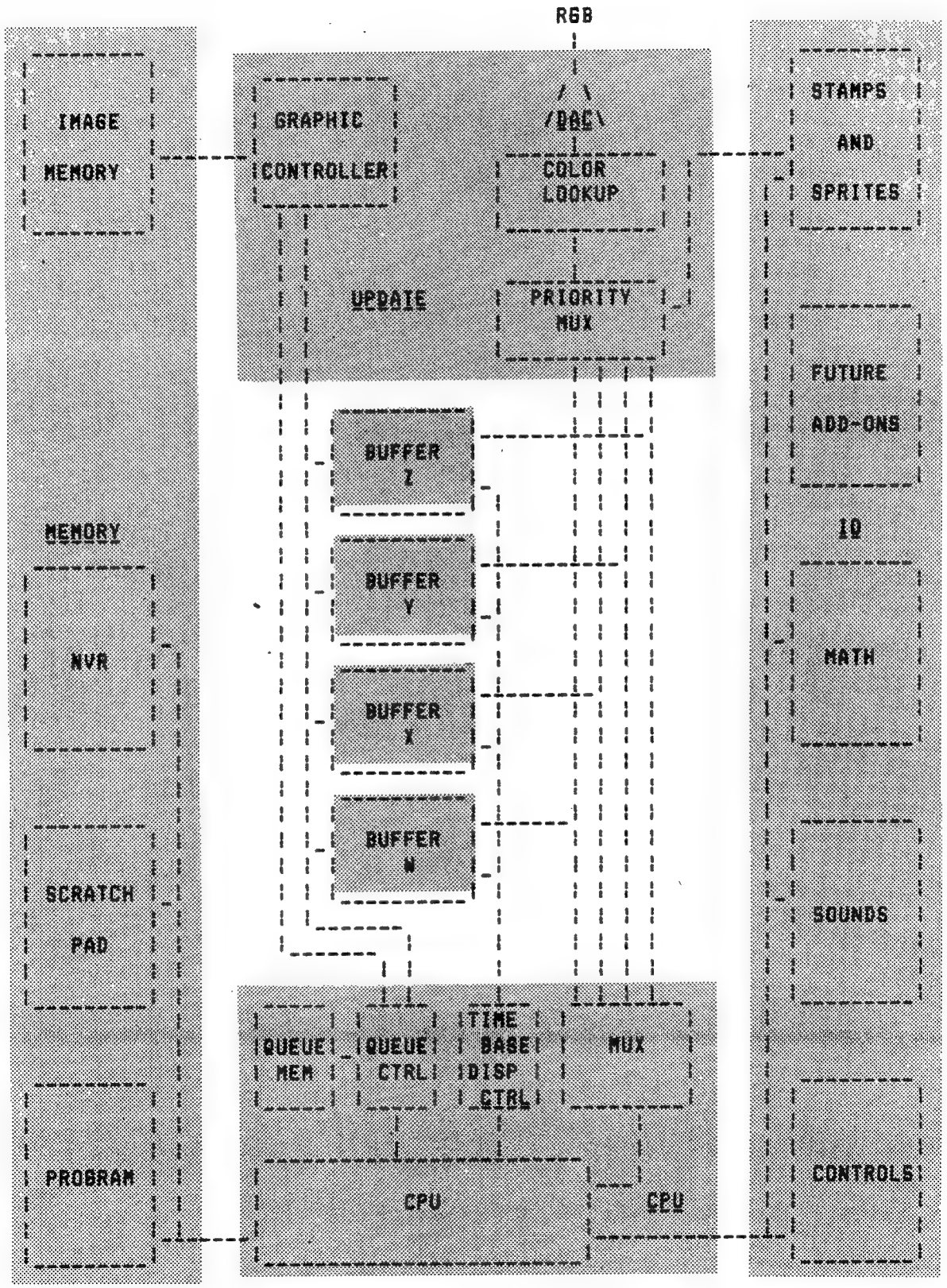
System IV hardware is modularly organized into 2 major components designated the Processor Module (PM) and Application Module (AM). The general purpose PM contains the CPU, QM, UC, SM and display logic. The game specific AM contains program memory, Image Memory, player IO, sound effects, and other game specific hardware as may be optionally desired.

CPU

The CPU is a 8 MHz 68000 or 68010 microprocessor with 16 megabyte direct addressing. The lower half of this memory is allocated to the AM and optional DM (Debug Module) while the upper half of CPU memory is entirely allocated to the PM and contains the Queue Memory, PM Input-Output Registers and Screen Memory.

[interrupts, power reset]

Detailed Block Diagram



Queue Memory

The Queue Memory is a dual port RAM that can be accessed by both the CPU and the Queue Processor under direction of the Queue Controller. Queue Memory provides the CPU with a command table to which high level character and vector commands can be issued. The Queue Processor reads and decodes the contents of Queue Memory when enabled.

Queue Controller

The Queue Controller interfaces the relatively slow CPU to the high speed Graphic Controller either directly or via the Queue Memory using the Queue Processor. The Queue Controller responds to all CPU accesses in the PM address space (800000H through FFFFFFH) which includes the Queue Memory, PM Input-Output Registers, and Screen Memory. Queue Controller operation is determined primarily by the CPU address and the <qp> bit of the Master Control Register.

The QC decodes the CPU address space via a 512 by 8 bit mapping PROM (PMMAP) that distinguishes Queue Memory, Control Register, Queue Register, Status Register and Screen Space address ranges.

[more to be added]

Graphic Controller

The Graphic Controller is a special purpose micro-coded processor that accepts commands issued directly by the CPU or fetched from Queue Memory by the Queue Processor to perform high speed update operations on Screen Memory and other System IV components. The Graphic Controller performs sophisticated multi-pixel image transfer, clipping, vector, and polygon fill operations as well as low level single byte/word read/write.

Screen Memory

The two to four 512 by 512 by 4 to 8 bit arrays used to hold images during update for output during display comprise the SM. All 512 lines are accessable during both update and display although only 384 lines (in any combination) may be output for display during any one frame. Screen Memory occupies an interior subset of Screen Space.

Color Memory

During display, the Screen Memory output is directed to the address input of Color Memory. Color Memory converts the 4 to 8 bits of pixel data from the highest priority active Buffer into from 4 to 8 bits each of red, blue, and

green. Each digital red, blue, and green value is then converted to the appropriate analog level by a high speed DAC that provides direct drive for the RGB display monitor.

Real-time modification of Screen Memory during display is performed by the Replacement Table portion of the Translation RAM of each Buffer assigned to display. This is accomplished using read-modify-write to channel output data back to input after processing through Translation RAM. The small delay between read and write is compensated by adjustment of the horizontal offset for the affected Buffers. Buffer input feedback can be selected from among any of the Buffers assigned to display. A Buffer may be assigned to replace into itself as well as any other Buffer assigned to display. Operations including copying, merging, swapping, or cycling Buffers are supported.

Image Transfer

The Image Transfer system consists of the Image Memory and mask devices located on the Memory board (ROM and PROM) and the Graphic Controller located on the Update board. The Graphic Controller generates the required character address for any given line and pixel of Image Memory. Pictures stored in Image Memory may occupy from 1 to 8 overlapping planes which may be grouped as desired to provide from 1 to 256 values per pixel character. Storage efficiency is optimized by combining images requiring less than 8 bits per pixel in the same byte of Image Memory. The mask hardware and shift hardware, together with the ALU of the Graphic Controller undoes this compression by extracting the desired contiguous bit range from the compacted 8 bits and mixing them with an 8 bit base value. This result is optionally passed through the Expansion Table of each Buffer assigned to update and the converted result is applied to the Screen Memory data input port.

Additional flexibility is provided by a comparator at the output of the mask/shift hardware that detects zero values in excess of a specified value. Depending upon the status of the image full/partial bit in the IFR, this result may be used to control writing to Screen Memory on a pixel by pixel basis. This permits the non-blank (non-zero) areas of pictures in Image Memory to be added to Screen Memory without adding the empty (zero) portions as well.

Finally, writing to Screen Memory may be limited to any combination of individual image planes in individual Buffers via the Plane Select Register and 2 control bits of the Buffers assigned to update.

Refresh

The dynamic memory used for Screen Memory in each Buffer requires refresh every 2 ms or 4 ms depending on the 64K RAM devices installed. Buffers assigned to display are refreshed 8 locations at time during horizontal retrace while Buffers assigned to update are refreshed 128 at a time at regular 2 ms intervals. Display refresh is completed every 32 lines and update refresh cycles every 48 lines for 128 location 2 ms devices and every 96 lines for 256 location 4 ms devices.

RAM Refresh Address Map

RAM row	A7	A6	A5	A4	A3	A2	A1	A0
Display	V0	H5	H4	H3	V4	V3	V2	V1
Update	R7	R6	R5	R4	R3	R2	R1	R0

Legend:

An	RAM row address bit n
Vn	Vertical timebase bit n
Hn	Horizontal timebase bit n
Rn	Refresh bit n

CPU Address to Screen Space Coordinate Mapping

CPU	Y/X
A23	1
A22	Y10
A21	Y09
A20	Y08
A19	Y07
A18	Y06
A17	Y05
A16	Y04
A15	Y03
A14	Y02
A13	Y01
A12	Y00
A11	X11
A10	X10
A09	X09
A08	X08
A07	X07
A06	X06
A05	X05
A04	X04
A03	X03
A02	X02
A01	X01
UDS-	X00

CPU Data to Graphic Register Hardware Mapping

D	Y	X	IH	IL
00	Y00	X00	IH00	IL00
01	Y01	X01	IH01	IL01
02	Y02	X02	IH02	IL02
03	Y03	X03	IH03	IL03
04	Y04	X04	IH04	IL04
05	Y05	X05	IH05	IL05
06	Y06	X06	IH06	IL06
07	Y07	X07	IH07	IL07
08	Y08	X08		IL08
09	Y09	X09		IL09
10	Y10	X10		IL10
11		X11		IL11
12				
13				
14				
15				

Schematic Drawings - Buffer

Each Buffer contains up to 32 64K dynamic RAMs supported by about 35 integrated circuits.

Each Buffer is linked to the rest of System IV by a number of common data buses. Each Buffer drives 1 of the 4 8-bit BDObn data buses used to output images during display and read data for the CPU during update. All Buffers receive input from all 4 BDObn buses. Timing control signals are received from the 8-bit DTCn bus during update and the 8-bit UTCn bus during display. Data for mode control is set while in update mode and received via the UCB bus. Address information used during display is received via the Vn bus. Data and address used during update are received via the Un bus.

A Buffer is placed in display mode whenever BCR signal BUENb- is set to 1. BUENb- forces the Un/Vn multiplexor (ICS xx and yy) to select addresses from the display address bus Vn. BUENb- also enables output from the display timing control register (IC xx) and disables the update timing control register (IC xx). Both Vn and DTCn are generated by the timebase circuitry. During each frame Vn provides sequential line counts from 0 to 383 together with higher frequency variations required for refresh. DTCn provide controls to perform RAS, CAS, and WE for the dynamic RAMs and other timing signals.

Once per visible line, Vn passes through the Un/Vn multiplexor and is loaded into both the Translation RAM Address Counter (TRAC) and Translation RAM Data Register (TRDR). On the next 16 MHz clock cycle (time reference 1 or TR1), the vertical address appears on both the TRAC and TRDR outputs. Bits 0 through 8 contain the vertical address

0 to 383. The TRAC, however, also has an additional 10th bit, TRA9, derived from the BDS8/9 multiplexor (IC xx). The BDS8/9 multiplexor routes input signal H0 during both TR0 and TR1. H0 is the lsb of the horizontal time base counter and toggles on each 16 MHz clock transition. H0 is 0 during TR0 and 1 during TR1. Therefore, TRA9 is 0 during TR1 and the full Translation RAM address comprised of bits TRA0 through TRA9 is in the 0 to 383 range. During TR2, TRA9 is 1 and the 10 bit address is in the 512 to 895 range.

Using 10 bit TRAn address, the Translation RAM (ICs XX, YY, and ZZ) provides a vertical offset during TR1 and a horizontal offset during TR2 on output bits TDRn. The vertical offset is loaded into the SRDR (IC xx) where it appears during TR2. During TR2, the vertical offset in SRDR is added via the Offset Adder (ICS xx and yy) to the vertical address. The vertical address is obtained from the 9 low bits of the TRAC.

[to be continued]

Hardware

IC Count

The following is an approximate IC distribution for the major circuit subsystems:

```
CPU, Queue, and Timebase
Update and Display
Buffer - minimum 2 required
 overhead                37
 4 RAMs per bit depth
                        =====
(37+4*<depth bits>)*<number of buffers>=
Memory
 Program Proms                              32
 Image Proms                                32
 Scratchpad
 Non-volatile memory
 Decoding
IO board
 Controls
 Sound effects
```

Configuration Component Requirement Guide

Buffers	Depth	RAMs	Buffer Total
2	4	32	106
2	5	40	114
2	6	48	122
2	7	56	130
2	8	64	138
3	4	48	155
3	5	60	167
3	6	72	179
3	7	84	191
3	8	96	203
4	4	64	204
4	5	80	220
4	6	96	236
4	7	112	252
4	8	128	268

Screen RAM Component Layout

An modulo 4=3	An modulo 4=2	An modulo 4=1	An modulo 4=0
longword 00 * oddword 00 lowbyte 0	longword 08 * oddword 08 highbyte 0	longword 16 * evenword 00 lowbyte 0	longword 24 * evenword 08 highbyte 0
01 * 01 1	09 * 09 1	17 * 01 1	25 * 09 1
02 * 02 2	10 * 10 2	18 * 02 2	26 * 10 2
03 * 03 3	11 * 11 3	19 * 03 3	27 * 11 3
04 * 04 4	12 * 12 4	20 * 04 4	28 * 12 4
05 * 05 5	13 * 13 5	21 * 05 5	29 * 13 5
06 * 06 6	14 * 14 6	22 * 06 6	30 * 14 6
longword 07 * oddword 07 lowbyte 7	longword 15 * oddword 15 highbyte 7	longword 23 * evenword 07 lowbyte 7	longword 31 * evenword 15 highbyte 7

Notes: An = address
* = pin 1 orientation marking on 64K RAM DIP package
lowbyte = all 8 bit bytes beginning at address An modulo 2 = 0
highbyte= all 8 bit bytes beginning at address An modulo 2 = 1
evenword= all 16 bit words beginning at address An modulo 4 = 0
oddword = all 16 bit words beginning at address An modulo 4 = 2
longword= all 32 bit longwords beginning at address An modulo 4 = 0

Glossary of Terms

To avoid confusion among often multiple definitions of common terms in the graphic, electronic, and software fields, and to give the authors a chance to redefine the language, this glossary of System IV terms is presented.

Buffer	4 to 8 image planes with common address offset values, display/update assignment, display output priority, display replacement routing, etc.
character	A set of stored points to be transferred from Image Memory to Screen Memory, i.e., a picture as opposed to a vector or polygon.
clipping	The process of limiting write access in the 4096 by 1536 Screen Space area to the 512 by 384 wide subset occupied by visible Screen Memory.
Color Memory	The three individual Color RAMs that form a 1024 by 24 bit look up table used to assign pixels to viewable colors.
Color RAM	One of three 1024 by 8 bit RAMs used as a lookup table to assign a pixel to one of the three video DACs to provided 256 levels per for each primary color.
Color Table	Any one of the 12 256 x 8 bit subranges of the Color Memory. There are 4 Color Tables, designated 0 through 3 for each of the 3 Color RAMs R, G, and B. The display priority scheme can be set to associate one or more Buffers with a particular Color Table.
display	Reserved to refer to operations involving output of Screen Memory contents to the CRT monitor.
Expansion Table	A 256 by 8 bit subrange of the Translation RAM used to decompress or convert the packed data values taken from Image Memory to an 8 bit value for transfer to Screen Memory. The Expansion Table timeshares the same physical memory space with the Replacment Table.

frame	The time required to display the entire contents of one Buffer and to perform vertical retrace for the display monitor. This is 1/60th second in System IV.
Graphic Controller	The micro-controller that executes commands issued by the Queue Controller. The Graphic Controller controls the timing of all update operations involving Screen Memory and Image Memory.
Horizontal Offset Table	The 384 by 9 bit subrange of the Translation RAM in each Buffer used to provide a displacement value of from 0 to 511 along the X axis for each line of Screen Memory during both update and display.
image	Pixels arranged to form a picture.
Image Memory	Up to 1 megabyte of PROM/ROM used to hold compressed pictures suitable for high speed transfer to Screen Memory.
image transfer	The process of reading a rectangular array of compressed pictures from Image Memory, selecting the desired bits from each pixel through mask and shift, merging with the IDR, converting through the Expansion Table, and writing the results to Screen Memory while maintaining proper address pointers to both Image and Screen Memory, performing clipping, and running at 16 MHz.
Offset Table	Either of two 384 by 9 bit subranges of the Translation RAM of each Buffer providing displacements for each visible line of Screen Memory during both update and display. Displacements are added to the respective horizontal and vertical Screen Memory coordinates to produce the actual coordinates supplied to Screen RAM. The Offset Tables makes possible high speed scrolling of Buffer images along either axis, compensation for replacement induced scrolling, vertical inversion of the display, and a variety of other tricks.
pixel	An individual spot on the screen. System IV has 512 pixels per line on 384 lines.
plane	A rectangular bit-mapped memory array 1 bit deep. System IV Buffers each contain from 4 to 8 planes addressed as a single array but individually write-enabled.
Queue Controller	The hardware that arbitrates access of Queue

	Memory by the CPU and Queue Processor when the Queue Processor is on and that passes CPU commands directly to the Graphic Controller when the Queue Processor is off.
Queue Memory	The dual port 16 bit wide memory used primarily to store CPU commands until fetched by the Queue Processor for use by the Graphic Controller.
Queue Processor	A hardware controller that sequentially fetches instructions from Queue Memory and loads them into the appropriate Control Registers.
Replacement Table	A 256 by 8 bit subrange of the Translation RAM of each Buffer used during display to convert pixels read from Screen Memory to new values written back to Screen Memory 4 horizontal locations later. The Replacement Table may be programmed to provide selective or general erase as well as other features on a frame by frame basis. The Replacment Table timeshares the same physical memory space with the Expansion Table.
Screen Memory	One or more of the bit-mapped Screen RAM arrays used to accumulate images during update for output during display. Screen Memory is an interior subset of Screen Space.
Screen RAM	The 512 by 512 by 4 to 8 bit deep random access memory array in each Buffer.
Screen Space	A 4096 by 1536 array of mostly write-only memory and Screen Memory used as the holding area for created by the CPU and Graphic Controller. The use of Screen Space to surround Screen Memory facillitates fast clipping of images.
scrolling	The process of horizontally or vertically translating an entire Buffer accomplished by adding a uniform offset to the horizontal and/or vertical coordinates. Scrolling may be performed on a line by line basis.
shadow read	A technique in which the contents of a hardware register is seemingly read by actually reading the contents of an associated RAM previously written using identical data.
source image	Picture data generally stored in Image ROM and compressed into the minimum number of bits required for the number of colors needed. Source image data is usually passed through

	the Expansion Table of the Translation RAM and then written to Screen Memory during update.
table	A subrange of a RAM used for lookup. System IV provides 4 Color Tables for each of 3 Color RAMs, 1 Vertical Offset Table, 1 Horizontal Offset Table, and 1 Expansion/Replacment Table for each of 4 Buffers.
Translation RAM	The 1K by 9 bits of fast Random Access Memory in each Buffer that performs Expansion during update, Replacement during display, and address offset during both update and display. The Translation RAM is divided into 3 sections that provide (1) 384 9-bit vertical offset values for each displayable line, (2) 384 9-bit horizontal offsets values for each displayable line, and (3) 256 bytes of user defined conversion tables for expansion, replacement, or transfer storage.
update	Reserved to refer to operations involving access of Screen Memory/Screen Space for the primary purpose of inputting original data in particular and any Buffer not set to display in general.
Vertical Offset Table	The 384 by 9 bit subrange of the Translation RAM in each Buffer used to provided a displacement value of from 0 to 511 along the Y axis for each line of Screen Memory during both update and display.
visible screen Memory	The portion of Screen Memory that is output during display. If the vertical Offset Table is set to 0's, this is the first 384 lines of the 512 by 512 Screen Memory array(s).
watchdog	Hardware that resets the CPU to a power-up state when- ever certain software operations fail to occur within a limited time frame.
windowing	Limiting input or output access of a character image to a specified rectangular subset.
zoom	Enlarging a specified portion of a source image character to fit a specified portion of screen memory during image transfer.

Appendices

Appendix A

Processor Module IO Registers

Queue Registers

Name	Address	Opcode D15--D08	Format	
GR0	FF8000	1000....	8!<GR0>	Graphic Register 0
GR1	FF8002	1001....	9!<GR1>	Graphic Register 1
GR2	FF8004	1010....	A!<GR2>	Graphic Register 2
GR3	FF8006	1011....	B!<GR3>	Graphic Register 3
GR4	FF8008	1100....	C!<GR4>	Graphic Register 4
GR5	FF800A	1101....	D!<GR5>	Graphic Register 5
GR6	FF800C	1110....	E!<GR6>	Graphic Register 6
GR7	FF800E	11110000	F0!<GR7>	Graphic Register 7
BCRW	FF8010	11110000	F0!<BCRW>	Buffer Control Register W
BCRX	FF8012	11110001	F1!<BCRX>	Buffer Control Register X
BCRY	FF8014	11110010	F2!<BCRY>	Buffer Control Register Y
BCRZ	FF8016	11110011	F3!<BCRZ>	Buffer Control Register Z
PSRW	FF8018	11110100	F4!<PSRW>	Plane Select Register W
PSRX	FF801A	11110101	F5!<PSRX>	Plane Select Register X
PSRY	FF801C	11110110	F6!<PSRY>	Plane Select Register Y
PSRZ	FF801E	11110111	F7!<PSRZ>	Plane Select Register Z
BPRW	FF8020	11111000	F8!<BPRW>	Buffer Priority Register W*
BPRX	FF8022	11111001	F9!<BPRX>	Buffer Priority Register X*
BPRY	FF8024	11111010	FA!<BPRY>	Buffer Priority Register Y*
BPRZ	FF8026	11111011	FB!<BPRZ>	Buffer Priority Register Z*
DPR	FF8028	11111100	FC!<DPR>	Display Priority Register
CTR	FF802A	11111101	FD!<CTR>	Color Table Register
IFR	FF802C	11111110	FE!<IFR>	Image Format Register

ECR Register	FF802E	11111111	FF¦<ECR>	Execution Control
FAR	FF8030	.....	¦<FAR>	Fetch Address Register

* Indicates not currently implemented.

Control Registers

Name	Address	Opcode D15--D08	Format	
MCR Register	FF8040	.....	<MCR>	Master Control
QLR	FF8042	.....	<QLR>	Queue Length Register
QAR	FF8044	.....	<QAR>	Queue Address Register
	FF8046	.....	< >	spare

Status Registers

Name	Address	Opcode D15--D08	Format	
DHR	FF8800	.....	<DAR>	Display Hold Register read only
DLR	FF8840	.....	<DAR>	Display Load Register sample and read

Appendix B

Buffer Priority Select Codes

DPR Value		Display Buffers	Buffer Priority			
Binary	Hex		1	2	3	4
00 0000	00	1	W	-	-	-
00 0001	01	2	W	X	-	-
00 0010	02	2	W	Y	-	-
00 0011	03	2	W	Z	-	-
00 0100	04	3	W	X	Y	-
00 0101	05	3	W	X	Z	-
00 0110	06	3	W	Y	X	-
00 0111	07	3	W	Y	Z	-
00 1000	08	3	W	Z	X	-
00 1001	09	3	W	Z	Y	-
00 1010	0A	4	W	X	Y	Z
00 1011	0B	4	W	X	Z	Y
00 1100	0C	4	W	Y	X	Z
00 1101	0D	4	W	Y	Z	X
00 1110	0E	4	W	Z	X	Y
00 1111	0F	4	W	Z	Y	X
01 0000	10	1	X	-	-	-
01 0001	11	2	X	W	-	-
01 0010	12	2	X	Y	-	-
01 0011	13	2	X	Z	-	-
01 0100	14	3	X	W	Y	-
01 0101	15	3	X	W	Z	-
01 0110	16	3	X	Y	W	-
01 0111	17	3	X	Y	Z	-
01 1000	18	3	X	Z	W	-
01 1001	19	3	X	Z	Y	-
01 1010	1A	4	X	W	Y	Z
01 1011	1B	4	X	W	Z	Y
01 1100	1C	4	X	Y	W	Z
01 1101	1D	4	X	Y	Z	W
01 1110	1E	4	X	Z	W	Y
01 1111	1F	4	X	Z	Y	W

DPR Value		Display	Buffer Priority			
Binary	Hex	Buffers	1	2	3	4
10 0000	20	1	Y	-	-	-
10 0001	21	2	Y	W	-	-
10 0010	22	2	Y	X	-	-
10 0011	23	2	Y	Y	-	-
10 0100	24	3	Y	W	X	-
10 0101	25	3	Y	W	Z	-
10 0110	26	3	Y	X	W	-
10 0111	27	3	Y	X	Z	-
10 1000	28	3	Y	Z	W	-
10 1001	29	3	Y	Z	X	-
10 1010	2A	4	Y	W	X	Z
10 1011	2B	4	Y	W	Z	X
10 1100	2C	4	Y	X	W	Z
10 1101	2D	4	Y	X	Z	W
10 1110	2E	4	Y	Z	W	X
10 1111	2F	4	Y	Z	X	W
11 0000	30	1	Z	-	-	-
11 0001	31	2	Z	W	-	-
11 0010	32	2	Z	X	-	-
11 0011	33	2	Z	Y	-	-
11 0100	34	3	Z	W	X	-
11 0101	35	3	Z	W	Y	-
11 0110	36	3	Z	X	W	-
11 0111	37	3	Z	X	Y	-
11 1000	38	3	Z	Y	W	-
11 1001	39	3	Z	Y	X	-
11 1010	3A	4	Z	W	X	Y
11 1011	3B	4	Z	W	Y	X
11 1100	3C	4	Z	X	W	Y
11 1101	3D	4	Z	X	Y	W
11 1110	3E	4	Z	Y	W	X
11 1111	3F	4	Z	Y	X	W

Appendix C

Graphic Controller Instruction Summary

Mnemonic	GR0	GR1	GR2	GR3	GR4	GR5	GR6
notes							
FXOT	start	size	data	step			
FYOT	start	size	data	step			
FERT	start	size	data	step			
* SMER	X	Y	start	size			
* ERSM	X	Y	start	size			
FCT	start	size	data	step	RGB+CT		
* SMCT	X	Y	start	size	RGB+CT		
* CTSM	X	Y	start	size	CT		
* SEXP							
* REXP							
* SOFF							
* ROFF							
* STPS							
* RTPS							
LDSR	data						
SIDR	data						
RCPW							
SCPW	LM	TM	RM	BM			
IMSM	X	Y	W	H	IL	IH	IO
IMSMPC	X	Y	W	H	IL	IH	IO
APP	X	Y	data				
RPP	DX	DY	data				
AVP	X	Y	data				
RVP	DX	DY	data				
APFS	X	Y	data				
APFV	X	Y	data				

```
  APFE          X       Y       data
* RPFS          DX      DY      data
* RPFV          DX      DY      data
* RPFE          DX      DY      data
-------------------------------------------------------------------------
-------
  POKE
-------------------------------------------------------------------------
-------

*       = not presently implemented


Notes:  1. requires screen write enable in BCR
        2. requires translation write enable in BCR
        3. utilizes IFR
        4. utilizes PSR
        5. utilizes programmable clipping window
        6. saves end coordinates
        7. utilizes previously saved end coordinates
        8. utilizes expansion and offset status bits
        9. utilizes IDR

Legend: CT      = color table selection in bits ......XX
        DX      = signed horizontal offset
        DY      = signed vertical offset
        H       = height of data block written to screen
        IH      = 8 high bits of source image address
        IL      = 12 low bits of source image address
        IO      = line to line offset of source image
        RGB     = color RAM write enables in bits .BGR....
        size    = length of data transfer
        step    = data value increment
        W       = width of data block written to screen
        X       = screen space X range 000 - 7FF, 400 - 5FF visible
        XLM     = screen left margin
        XRM     = screen right margin
        Y       = screen space Y range 000 - 5FF, 200 - 37F visible
        YBM     = screen bottom margin
        YTM     = screen top margin
```

Appendix D

Image Format Register Mask and Shift Values

Mask Values

Mask Code	Active Image Data Bits	Comments
0	....	All bits off (forced to zero)
1	X	Image bit 0 only, all others forced to zero
2	XX	Image bits 0 and 1 only, all others forced to zero
3	XXX	etc.,etc.
4	XXXX	
5	...XXXXX	
6	..XXXXXX	
7	.XXXXXXX	
8	XXXXXXXX	All bits on (pass image data intact)
9	XXXXXXX.	All bits on except bit 0
A	XXXXXX..	All bits on except bits 0 and 1
B	XXXXX...	etc.,etc.
C	XXXX....	
D	XXX.....	
E	XX......	
F	X.......	Image bit 7 only, all others forced to zero

Legend:
X = active bit position (pass image data intact)
. = inactive bit position (zero substituted for image)

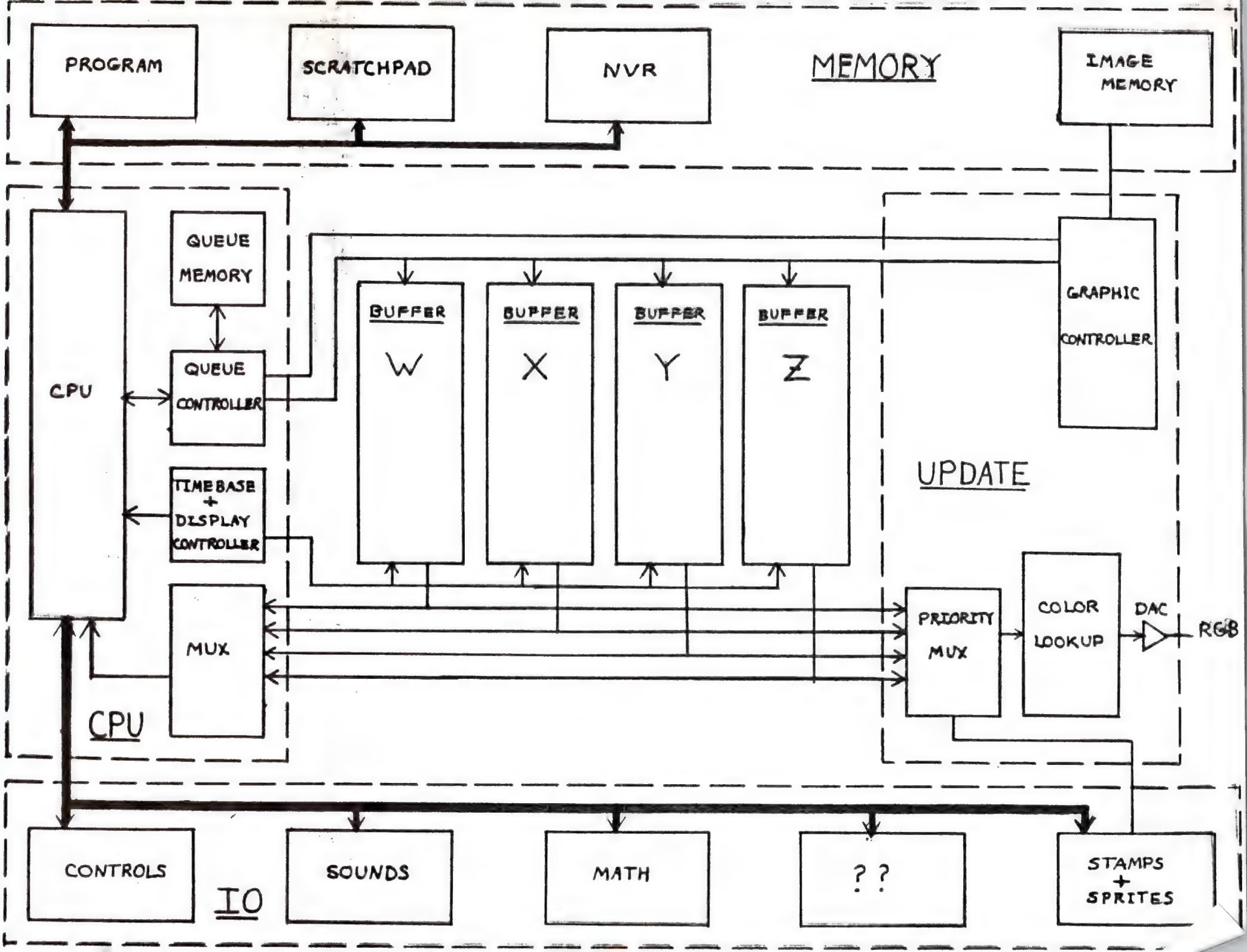
Shift Values

Shift Code	Image Data Position	
0	76543210	No masked output bits shifted
1	.7654321	Shift in 1 zero
2	..765432	Shift in 2 zeros
3	...76543	
4	7654	
5	765	
6	76	
7	7	Masked output bit 7 shifted to bit 0 position

Legend:
. = inactive bit position (image data forced to

zero)

7,6,5,4,3,2,1,0 = Image Memory output bits from mask hardware


PROGRAM
SCRATCHPAD
NVR
MEMORY
IMAGE MEMORY
CPU
QUEUE MEMORY
QUEUE CONTROLLER
TIMEBASE + DISPLAY CONTROLLER
MUX
CPU
BUFFER W
BUFFER X
BUFFER Y
BUFFER Z
GRAPHIC CONTROLLER
UPDATE
PRIORITY MUX
COLOR LOOKUP
DAC
RGB
CONTROLS
IO
SOUNDS
MATH
? ?
STAMPS + SPRITES

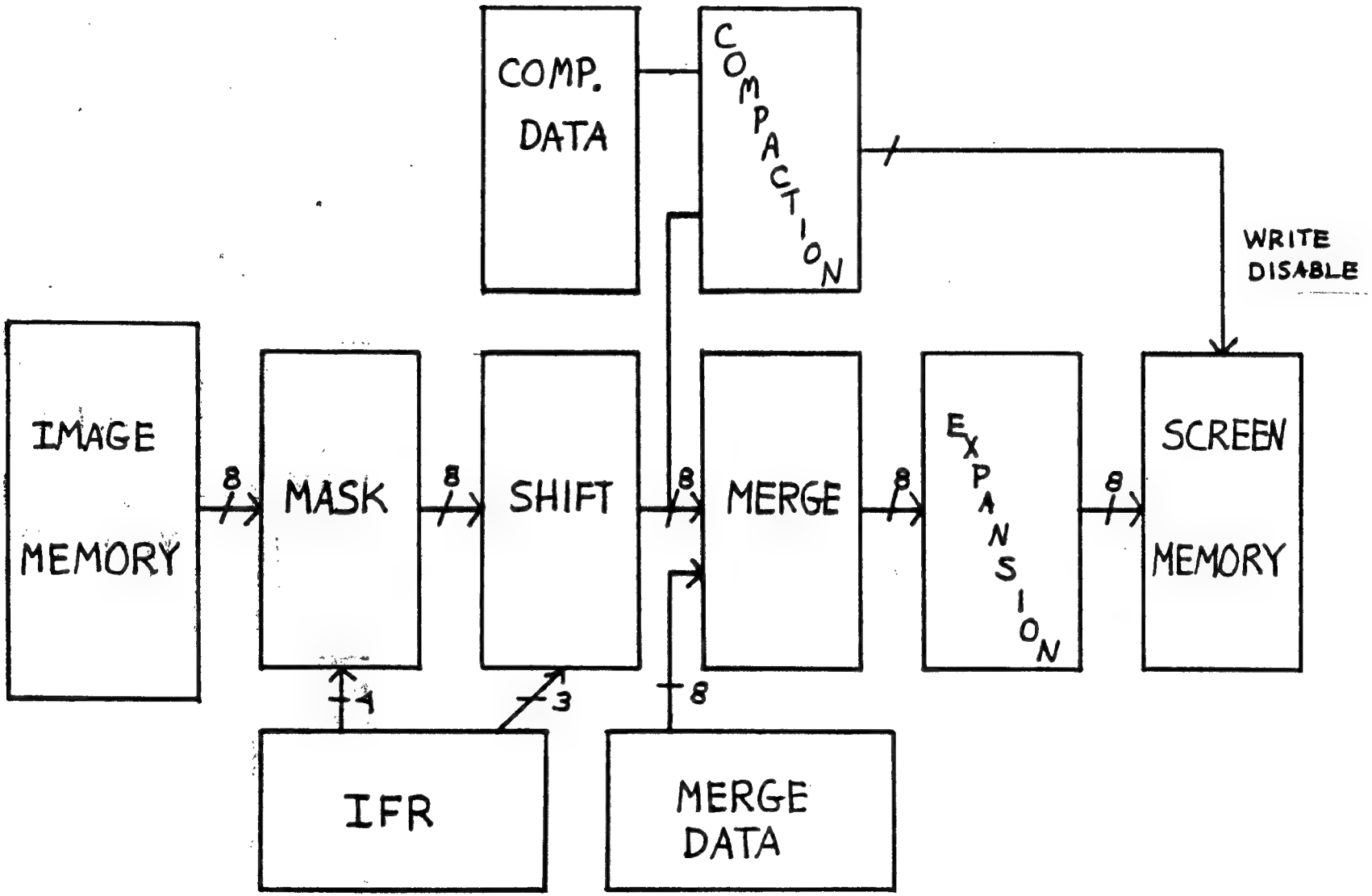
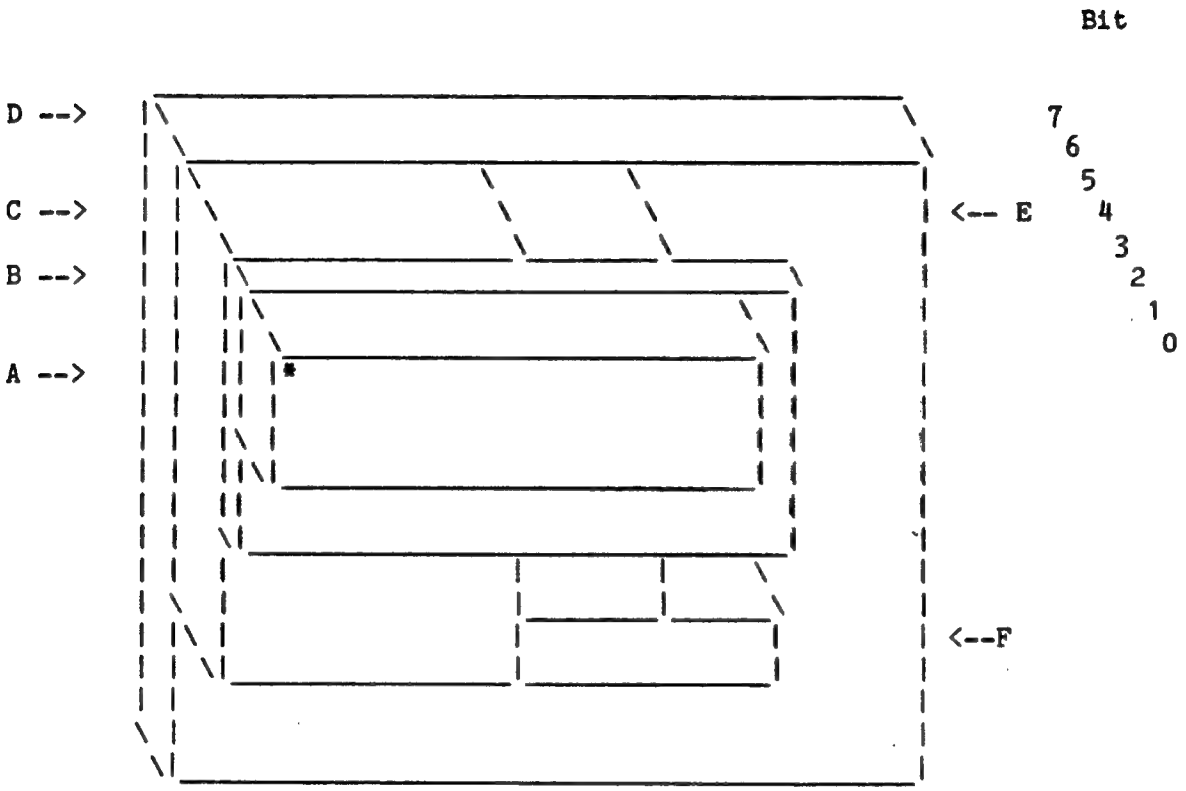


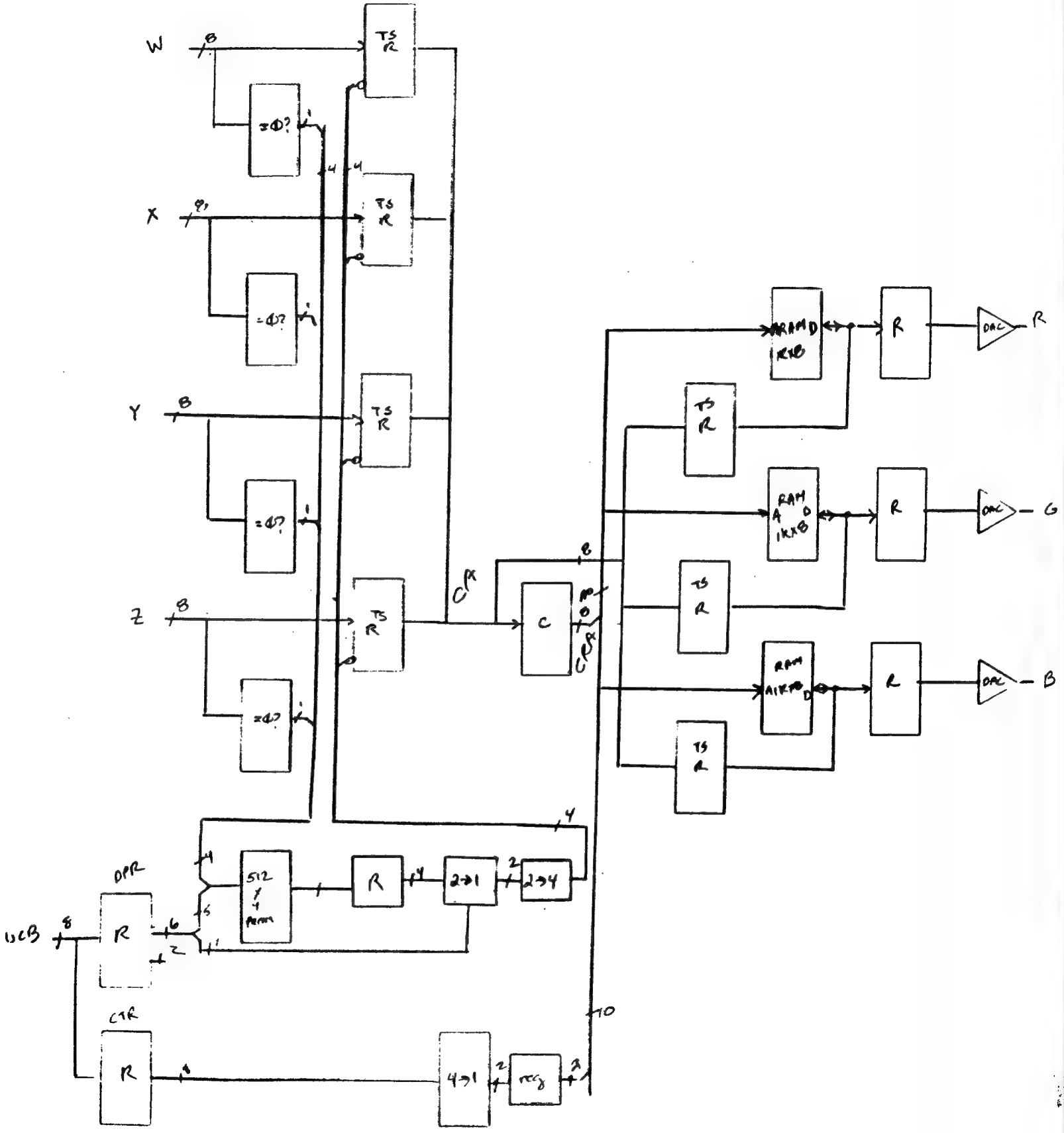
IMAGE TRANSFER PATH

Sample Image Memory Storage

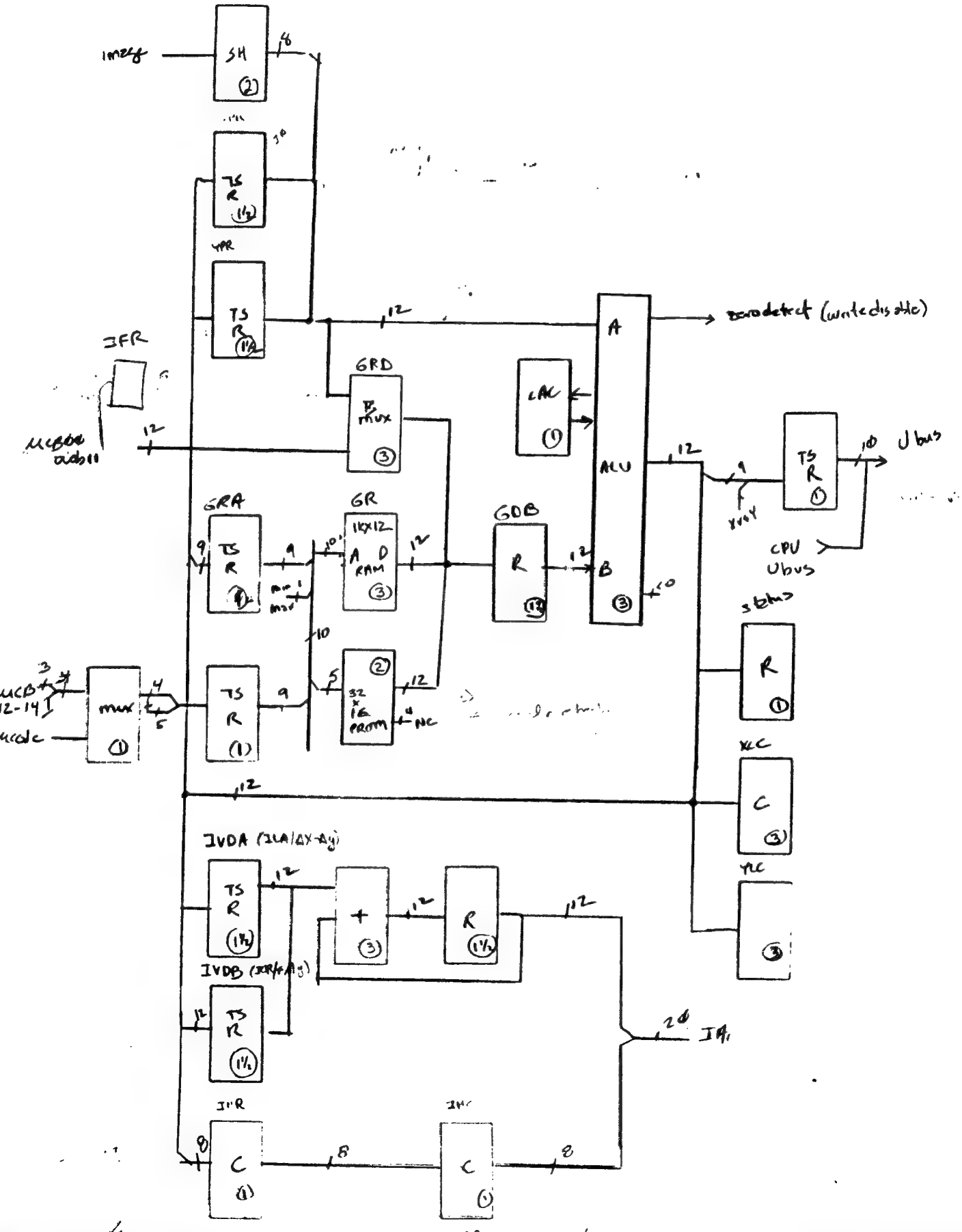


8309.22

Display

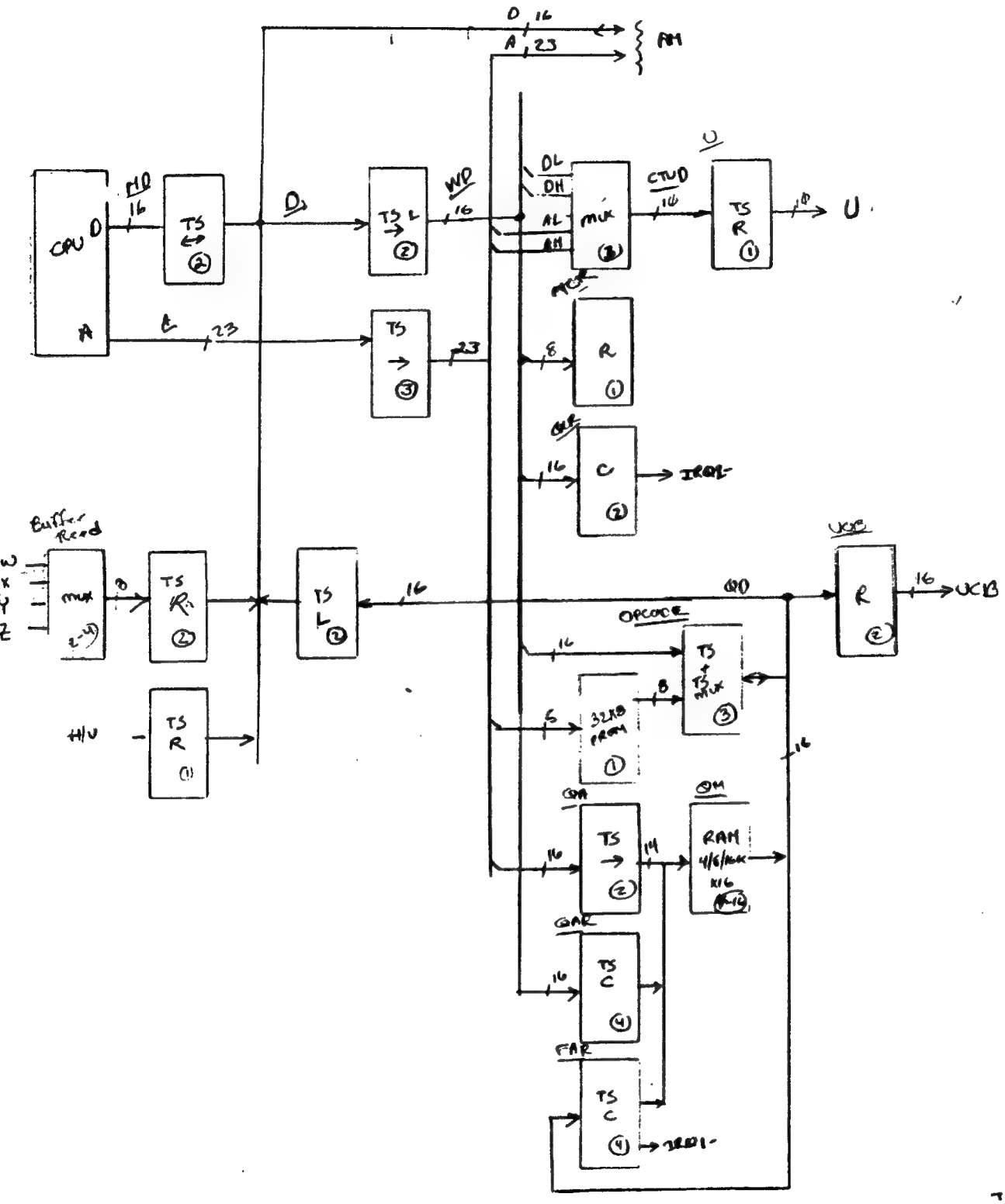


Graphic Controller Block Diagram



(M) + UC Block Diagram

8401.13
8401.15
8401.16
8401.19
8401.20
8402.09
8402.22



TS = Tri state
R = register
L = latch
mux = multiplexer
C = counter

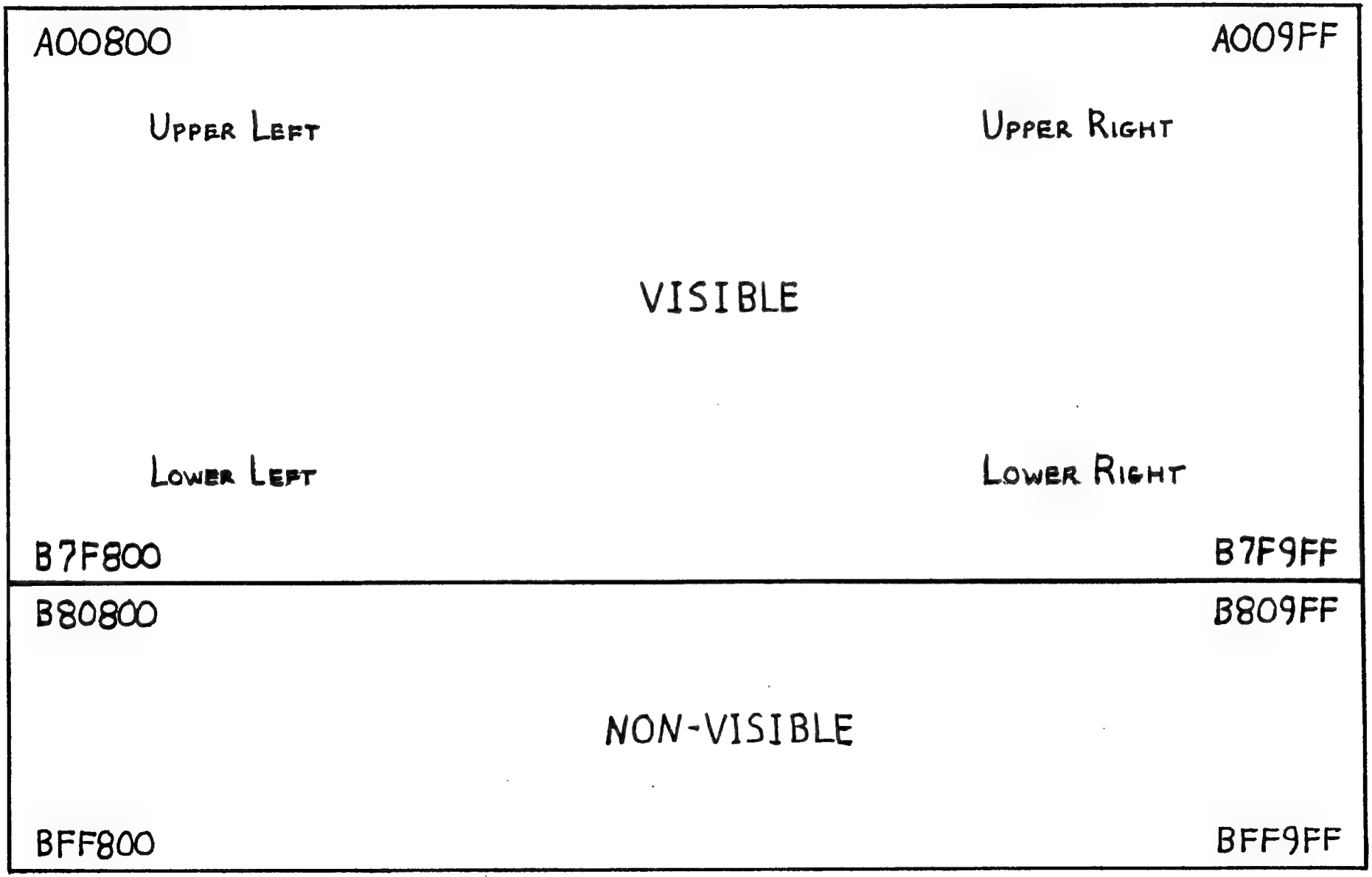
Layout of Screen Space

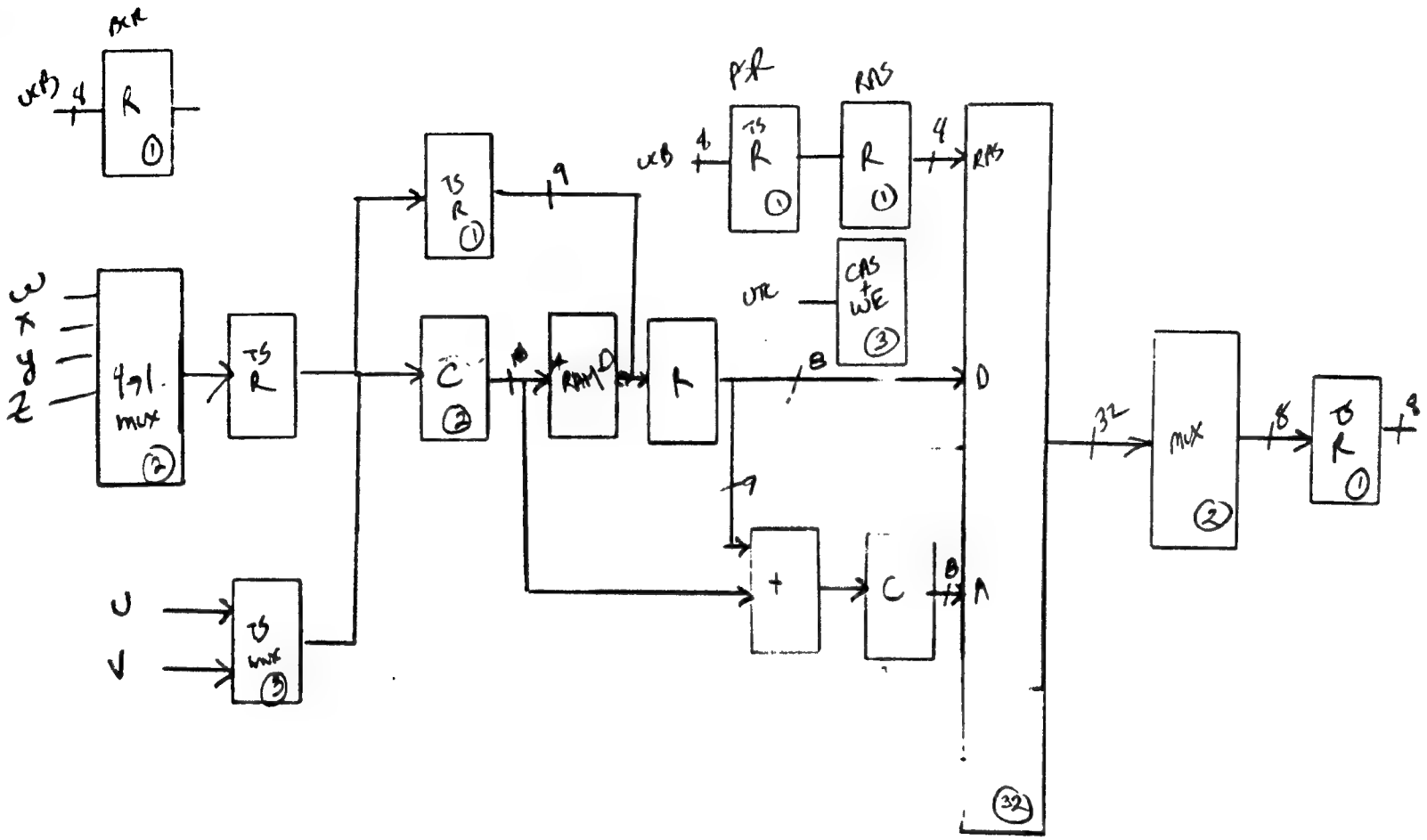
<table>
<tr><td>800000 800?FF
Upper Left
9FF000 9FF7FF</td><td colspan="2">800800 800FFF
Upper Right
9FF800 9FFFFF</td></tr>
<tr><td>A00000 A007FF

BFF000 BFF7FF</td><td>A00800 A009FF
Screen Memory
BFF800 BFF9FF</td><td>A00C00 A00FFF

BFFC00 BFFFFF</td></tr>
<tr><td>C00000 C007FF
Lower Left
DFF000 DFF7FF</td><td colspan="2">C00800 C00FFF
Lower Right
DFF800 DFFFFF</td></tr>
</table>

Layout of Screen Memory





Buffer Block Diagram

Color Memory

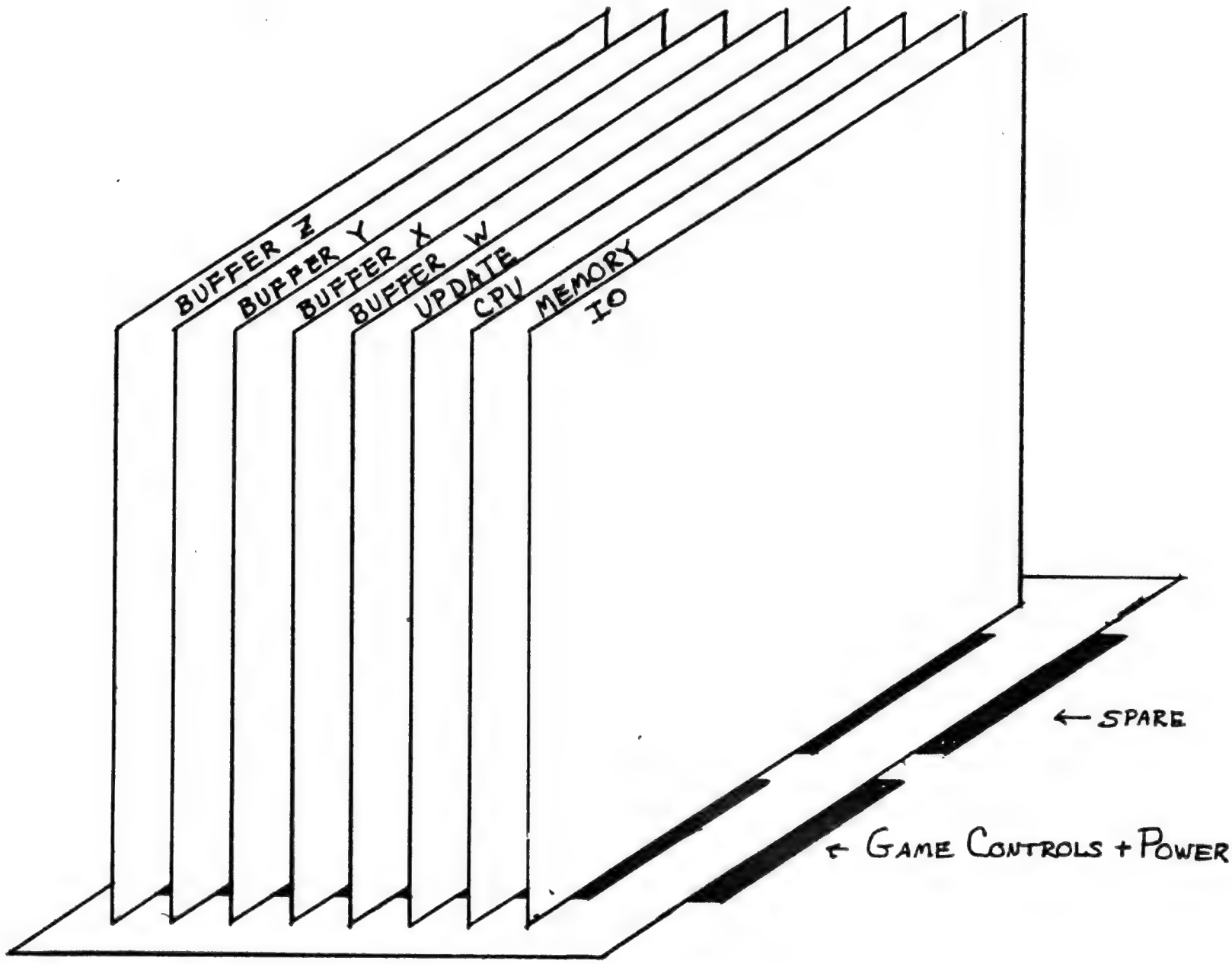
Color Table Number	Address Range		
	Red Color RAM	Green Color RAM	Blue Color RAM
0	000H to 0FFH	000H to 0FFH	000H to 0FFH
1	100H to 1FFH	100H to 1FFH	100H to 1FFH
2	200H to 2FFH	200H to 2FFH	200H to 2FFH
3	300H to 3FFH	300H to 3FFH	300H to 3FFH

Translation RAM Address to Function Map

RAM Address	Contents
000H	Vertical Offset for line 000H (000)
....	
17FH	Vertical Offset for line 17FH (383)
180H	Replacement/Expansion for data 00H
....	
1FFH	Replacement/Expansion for data 7FH
200H	Horizontal offset for line 000H (000)
....	
37FH	Horizontal offset for line 17FH (383)
380H	Replacement/Expansion for data 80H
....	
3FFH	Replacement/Expansion for data FFH

Translation RAM Function to Address Map

	Screen Line	Storage Address
Vertical Offset	000H (000)	000H
	.....	....
	17FH (383)	17FH
Horizontal Offset	000H (000)	200H
	.....	....
	17FH (383)	37FH
	Input Data	Storage Address
Expansion/Replacement	00H	180H
	...	....
	7FH	1FFH
	80H	380H
	...	....
	FFH	3FFH



SYSTEM IV PACKAGING

Set #2
orange label

Graphic Controller Instruction Summary

	Mnemonic	GR0	GR1	GR2	GR3	GR4	GR5	GR6	notes
E0	FXOT =17 48 18	start	size	data	step				
E1	FYOT =15 46 16	start	size	data	step				
E2	FERT =1D	start	size	data	step				
E3 *	SMER	X	Y	start	size				
E4 *	ERSM	X	Y	start	size				
E5	FCT =D3 4F 1F	start	size	data	step	CT+RGB			
F6 *	SMCT	X	Y	start	size	CT+RGB			
E7 *	CTSM	X	Y	start	size	CT			
D0 *	SEXP								
1 *	REXP								
2 *	SOFF								
3 *	ROFF								
4 *	STPS								
5 *	RTPS								
6	LDSR =10 11	data							
7	SIDR	data							
8	RCPW	Xmin	Ymin	Rmax	Ymax				
9	SCPW 40	LM	TM	RM	BM				
F0	IMSM 59	X	Y	W	H	IL	IH	IO	
F1	IMSMPC C4 (46)	Y	W	H	IL	IH	IO		
F2	PPA =C4 F8	X	Y	data					code started = 2C
F3	PPR =C5 F9	X	DY	data					
F4	AVPA =A4 C6	X	Y	data					
F5	RVPR =A7 D1	DX	DY	data					
F6	AFS A	X	Y	[redacted]					code started = 148 #2
F7	AFV A	X	Y	[redacted]					
F8	AFE A	[redacted]	[redacted]	data					closes
FA *	RFS R	DX	DY	[redacted]					
FB *	RFV R	DX	DY	[redacted]					
FC *	RFE R	[redacted]	[redacted]	data					
	POKE =12 13								

Free space
End 19C

* = not presently implemented

100 x 52

init = 0
sm1 = 2 (code started = 4)

Next

Notes:
1. requires screen write enable in BCR
2. requires translation write enable in BCR
3. utilizes IFR
4. utilizes PSR
5. utilizes programmable clipping window
6. saves end coordinates
7. utilizes previously saved end coordinates
8. utilizes expansion and offset status bits
9. utilizes IDR

Legend:

CT	= color table selection in bits XX
DX	= signed horizontal offset
DY	= signed vertical offset
H	= height of data block written to screen
IH	= 8 high bits of source image address
IL	= 12 low bits of source image address
IO	= line to line offset of source image
RGB	= color RAM write enables in bits .BGR....
size	= length of data transfer
step	= data value increment
W	= width of data block written to screen
X	= screen space X range 000 - 7FF, 400 - 5FF visible
XLM	= screen left margin
XRM	= screen right margin
Y	= screen space Y range 000 - 5FF, 200 - 37F visible
YBM	= screen bottom margin
YTM	= screen top margin

```
.TITLE  INSMC   - MC68000 ASSEMBLER TEST
.SBTTL INSMC    - MC68000 ASSEMBLER TEST
.IDENT  /31MAY0/

.NLIST  TTM
.LIST   BEX
.NLIST  HEX
.RADIX 10.

;ADDRESS MODES

        .NTYPE  TP,D0           ;DATA REGISTER DIRECT
        .NTYPE  TP,A3           ;ADDRESS REGISTER DIRECT
        .NTYPE  TP,(A4)         ;REGISTER INDIRECT
        .NTYPE  TP,(A5)+        ;AUTO INCREMENT
        .NTYPE  TP,-(A6)        ;AUTO DECREMENT
        .NTYPE  TP,4(A7)        ;REGISTER INDIRECT WITH DISPLACEMENT
        .NTYPE  TP,5(A3,D2)     ;REGISTER INDIRECT WITH INDEX
        .NTYPE  TP,5(A3,W^D2)   ;REGISTER INDIRECT WITH WORD INDEX)
        .NTYPE  TP,5(A3,L^D2)   ;REGISTER INDIRECT WITH LONG INDEX
        .NTYPE  TP,@#5          ;ABSOLUTE SHORT ADDRESS
        .NTYPE  TP,W^@#5        ;ABSOLUTE SHORT ADDRESS
        .NTYPE  TP,L^@#5        ;ABSOLUTE LONG ADDRESS
        .NTYPE  TP,LABEL        ;PC RELATIVE
        .NTYPE  TP,5(D3)        ;PC RELATIVE INDEXED
        .NTYPE  TP,5(W^D3)      ;PC RELATIVE WITH WORD INDEX
        .NTYPE  TP,5(W^A3)      ;PC RELATIVE WITH WORD INDEX
        .NTYPE  TP,5(L^D3)      ;PC RELATIVE WITH LONG INDEX
        .NTYPE  TP,5(L^A3)      ;PC RELATIVE WITH LONG INDEX
        .NTYPE  TP,#5           ;IMMEDIATE
        .NTYPE  TP,CCR          ;CONDITION CODE REGISTER
        .NTYPE  TP,SR           ;STATUS REGISTER
        .NTYPE  TP,USP          ;USER STACK POINTER
        .NTYPE  TP,VBR          ;VECTOR BASE REGISTER(68010)
        .NTYPE  TP,SFC          ;SOURCE FUNCTION CODE REGISTER(68010)
        .NTYPE  TP,DFC          ;DESTINATION FUNCTION CODE REGISTER(68010)

;REGISTER LISTS

        .NTYPE  TP,D0/A1/D3
        .NTYPE  TP,D1-D4
        .NTYPE  TP,A0-A3/D1-D5/D7

;DEFINE SYMBOLS

SYM1    =       123*456 + <345*567>

TAB     =       ^O<11>
CR      =       ^O<15>
LF      =       ^O<12>

MASK    =       ^B<10110011>
```

Susan
408-946-3027

```
.macro  tzmdef

gr0     = %FF8000
gr1     = %FF8002
gr2     = %FF8004
gr3     = %FF8006
gr4     = %FF8008
gr5     = %FF800A
gr6     = %FF800C

bcrw    = %FF8010
psrw    = %FF8018
dpr     = %FF8028
ctr     = %FF802A
ecr     = %FF802E
mcr     = %FF8040

fxot    = %10           ;fill x offset table
fyot    = %11           ;fill y offset table
fct     = %15           ;fill color table
ldsr    = %0E           ;load status register
poke    = %0F           ;poke graphic memory
pfoa    = %28           ;poly fill open absolute
pfor    = %29           ;poly fill open relative
pfva    = %2A           ;vector for absolute plot
pfvr    = %2B           ;vector for relative plot
pff     = %2C           ;fill polygon
rcw     = %08           ;reset clipping window
scw     = %09           ;set clipping window

a_lm    = %3F4          ;left margin address
a_rm    = %3F5          ;right margin address

lm      = %400
rm      = %5FF

c_bgr   = ^b01110000
c___r   = ^b01000000
c__g_   = ^b00100000
c_b__   = ^b00010000
.endm
```

Graphic Controller Instruction Summary

UCODE.RND/8404.03/TZM from S4.RND

Code	GR0	GR1	GR2	GR3	GR4	GR5	GR6	notes
00 REXP								
01 SEXP								
02 ROFF								
03 SOFF								
04 RTPS								
05 STPS								
06 SID	data							9
07 SIC	data							
08 RCW								
09 SCW	LM	TM	RM	BM				
0A - 0D reserved								
0E LDSR	data							
0F POKE	A	data						
10 FXOT	start	size	data	step				2
11 FYOT	start	size	data	step				2
12*SMER	X	Y	start	size				2,3
13*ERSM	X	Y	start	size				1,3
14 FERT	start	size	data	step				2
15 FCT	start	size	data	step	BGR..CT			
16*SMCT	X	Y	start	size	BGR..CT			3
17*CTSM	X	Y	start	size	CT			1,3
18 - 1F reserved								
20 ITSN	X	Y	W	H	IL	IH	IO	1,3,5,9
21*ITSC	X	Y	W	H	IL	IH	IO	1,3,5,9
22*ITSF								
23 reserved								
24 PPA	X	Y	data					1,6
25 PPR	DX	DY	data					1,6
26*VPA	X	Y	data					1,6
27*VPR	DX	DY	data					1,6
28 PFOA	X	Y						6
29 PFOR	DX	DY						6
2A PFVA	X	Y						6,7
2B PFVR	DX	DY						6,7
2C*PFCA	X	Y	data					1,6,7,8
2D*PFCR	DX	DY	data					1,6,7

* = not installed

Notes:
1. requires screen write enable in BCR
2. requires translation write enable in BCR
3. utilizes IFR
4. utilizes PSR
5. utilizes programmable clipping window
6. saves end coordinates
7. utilizes previously saved end coordinates
8. utilizes expansion and offset status bits
9. utilizes IDR

Legend:

A	= graphic controller internal RAM address
CT	= color table selection in bits ......CT
DX	= signed horizontal offset
DY	= signed vertical offset
H	= height of data block written to screen
IH	= 8 high bits of source image address
IL	= 12 low bits of source image address
IO	= line to line offset of source image
BGR	= color RAM write enables in bits .BGR....
size	= length of data transfer
step	= data value increment
W	= width of data block written to screen
X	= screen space X range 000 - 7FF, 400 - 5FF visible
XLM	= screen left margin
XRM	= screen right margin
Y	= screen space Y range 000 - 5FF, 200 - 37F visible
YBM	= screen bottom margin
YTM	= screen top margin

Graphic Controller Instruction Summary

UCODE.RND/8404.09/TZM from S4.RND

Code	GR0	GR1	GR2	GR3	GR4	GR5	GR6	notes
00 REXP								
01 SEXP								
02 ROFF								
03 SOFF								
04 RTPS								
05 STPS								
06 SIMD	data							
07*SICD	data							
08 RCW								
09 SCW	LM	TM	RM	BM				
0A - 0D reserved								
0E LDSR	data							
0F POKE	A	data						
10 FXOT	start	size	data	step				2
11 FYOT	start	size	data	step				2
12*SMER	X	Y	start	size				2,4
13*ERSM	X	Y	start	size				1,4
14 FERT	start	size	data	step				2
15 FCT	start	size	data	step	BGR..CT			
16*SMCT	X	Y	start	size	BGR..CT			4
17 CTSM	X	Y	start	size	CT			1,3,4
18 - 1F reserved								
20 ITSN	X	Y	W	H	IL	IH	IO	1,3,4,5,9,10
21*ITSC	X	Y	W	H	IL	IH	IO	1,3,4,5,9,10
22*ITSF	X	Y	W	H	IL	IH	IO	1,3,4,9,10
23 reserved								
24 PPA	X	Y	data					1,4,6
25 PPR	DX	DY	data					1,4,6
26*VPA	X	Y	data					1,4,6
27*VPR	DX	DY	data					1,4,6
28 PFOA	X	Y						6
29 PFOR	DX	DY						6,7
2A PFVA	X	Y						6,7
2B PFVR	DX	DY						6,7
2C*PFCA	data							1,4,6,7,8
2D*PFCR	data							1,4,6,7,8

* = not installed

Notes:
1. requires screen write enable in BCR
2. requires translation write enable in BCR
3. utilizes IFR
4. utilizes PSR
5. utilizes programmable clipping window
6. saves end coordinates
7. utilizes previously saved end coordinates
8. utilizes expansion and offset status bits
9. utilizes IMD
10. utilizes ICD

Legend:

A	= graphic controller internal RAM address
CT	= color table selection in bits ......CT
DX	= signed horizontal offset
DY	= signed vertical offset
H	= height of data block written to screen
IH	= 8 high bits of source image address
IL	= 12 low bits of source image address
IO	= line to line offset of source image
BGR	= color RAM write enables in bits .BGR....
size	= length of data transfer
step	= data value increment
W	= width of data block written to screen
X	= screen space X range 000 - 7FF, 400 - 5FF visible
XLM	= screen left margin
XRM	= screen right margin
Y	= screen space Y range 000 - 5FF, 200 - 37F visible
YBM	= screen bottom margin
YTM	= screen top margin

Graphic Controller Instruction Summary

Code	GR0	GR1	GR2	GR3	GR4	GR5	GR6	notes
00 REXP								
01 SEXP								
02 ROFF								
03 SOFF								
04 RTPS								
05 STPS								
06 SIMD	data							
07 SICD	data							
08 RCW								
09 SCW	LM	TM	RM	BM				
0A - 0D	reserved							
0E LDSR	data							
0F POKE	A	data						
10 FXOT	start	size	data	step				2
11 FYOT	start	size	data	step				2
12*SMER	X	Y	start	size				2,4
13*ERSM	X	Y	start	size				1,4
14 FERT	start	size	data	step				2
15 FCT	start	size	data	step	BGR..CT			
16 SMCT	X	Y	start	size	BGR..CT			4
17 CTSM	X	Y	start	size	CT			1,3,4
18 - 1F	reserved							
20 ITSN	X	Y	W	H	IL	IH	IO	1,3,4,5,9,10
21*ITSC	X	Y	W	H	IL	IH	IO	1,3,4,5,9,10
22*ITSF	X	Y	W	H	IL	IH	IO	1,3,4,9,10
23	reserved							
24 PPA	X	Y	data					1,4,6
25 PPR	DX	DY	data					1,4,6
26*VPA	X	Y	data					1,4,6
27*VPR	DX	DY	data					1,4,6
28 PFOA	X	Y						6
29 PFOR	DX	DY						6,7
2A PFVA	X	Y						6,7
2B PFVR	DX	DY						6,7
2C PFC	data							1,4,6,7,8
2D	reserved							

* = not installed

Notes:
1. requires screen write enable in BCR
2. requires translation write enable in BCR
3. utilizes IFR
4. utilizes PSR
5. utilizes programmable clipping window
6. saves end coordinates
7. utilizes previously saved end coordinates
8. utilizes expansion and offset status bits
9. utilizes IMD
10. utilizes ICD

Legend:

A	= graphic controller internal RAM address
CT	= color table selection in bits ......CT
DX	= signed horizontal offset
DY	= signed vertical offset
H	= height of data block written to screen
IH	= 8 high bits of source image address
IL	= 12 low bits of source image address
IO	= line to line offset of source image
BGR	= color RAM write enables in bits .BGR....
size	= length of data transfer
step	= data value increment
W	= width of data block written to screen
X	= screen space X range 000 - 7FF, 400 - 5FF visible
XLM	= screen left margin
XRM	= screen right margin
Y	= screen space Y range 000 - 5FF, 200 - 37F visible
YBM	= screen bottom margin
YTM	= screen top margin

UCODE.RND/8404.18/TZM from S4.RND